1 · ENGLISH SECTION
2 · VERSION FRANÇAISE
3 · DEUTSCHE GRUPPE
4 · SEZIONE ITALIANO
5 · SECCION ESPAÑOL

1 VIDEO GAME
JEU VIDEO
VIDEO-SPIEL
VIDEO GIOCO
JUEGO VIDEO

BASIC PROGRAMMING

INTRODUCTION

BASIC PROGRAMMING is an instructional tool designed to teach you the fundamental steps of computer programming. **BASIC** is an acronym for **B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. It was originated so that people could easily learn to "write" computer programs.

Computer programs are simply a series of instructions. The programs control the flow of information within the computer. **BASIC PROGRAMMING** allows you to give the Video Computer System™ the instructions it needs to carry out some simple tasks.

Keep in mind that **BASIC PROGRAMMING** has a limited amount of memory in comparison to more sophisticated computer systems. It is, however, an excellent instructional device for learning the essentials of computer programming.

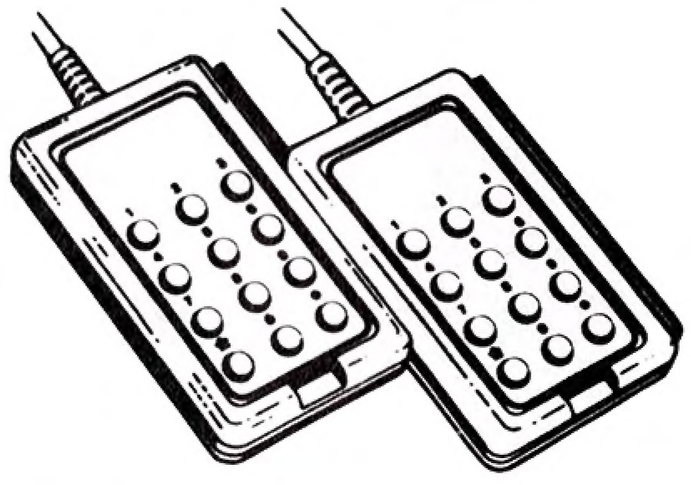
NOTE: Always turn the console **power** switch **off** when inserting or removing an ATARI Game Program™ cartridge. This will protect the electronic components and prolong the life of your ATARI Video Computer System game.

This Game Program may cause some television screens to "roll" slightly. If this occurs, adjustment of the VERTICAL HOLD may be necessary.

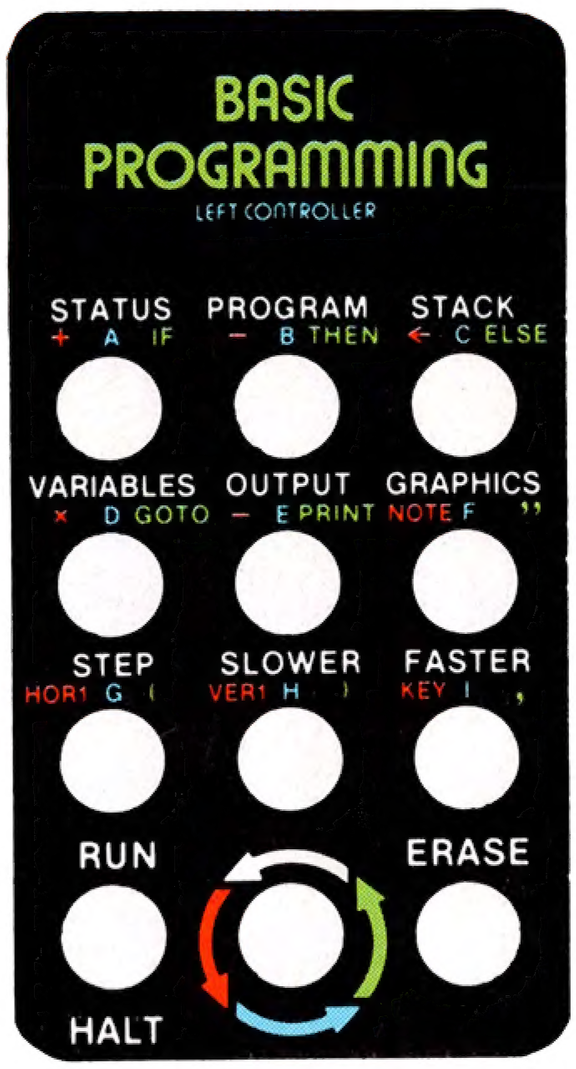
KEYBOARD CONTROLLERS

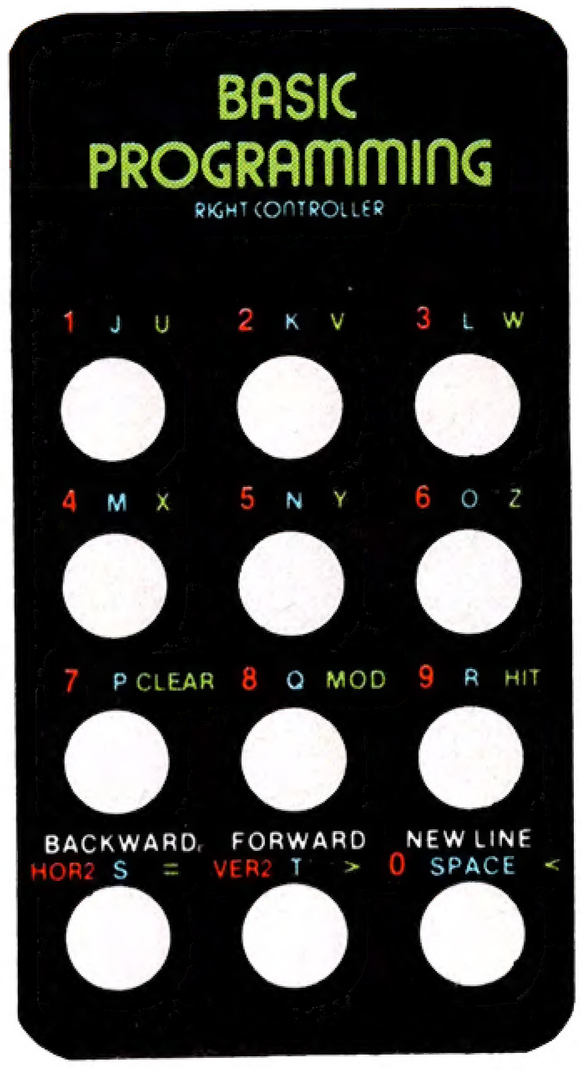
Use your Keyboard Controllers with this ATARI® Game Program™ cartridge. Be sure the Keyboard Controller cables are firmly plugged into the CONTROLLER jacks at the back of your ATARI Video Computer System™ game. *See Section 3 of your Owner's Manual for further details.*

To connect the two controllers, slide the tongue of the left controller into the groove on the right controller. The two Keyboard Controllers, locked together, form a 24-key Keyboard used to enter your program and to control the display on your television screen.



Remove the Keyboard Controller labels from the envelope. Place the label marked **LEFT** over the Keyboard plugged into the **LEFT CONTROLLER** jack at the rear of your computer console. Place the label marked **RIGHT** over the Keyboard plugged into the **RIGHT CONTROLLER** jack at the rear of your computer console.





DISPLAY REGIONS

The display is divided into six regions:

1. The **PROGRAM** region is used to give the computer your instructions.

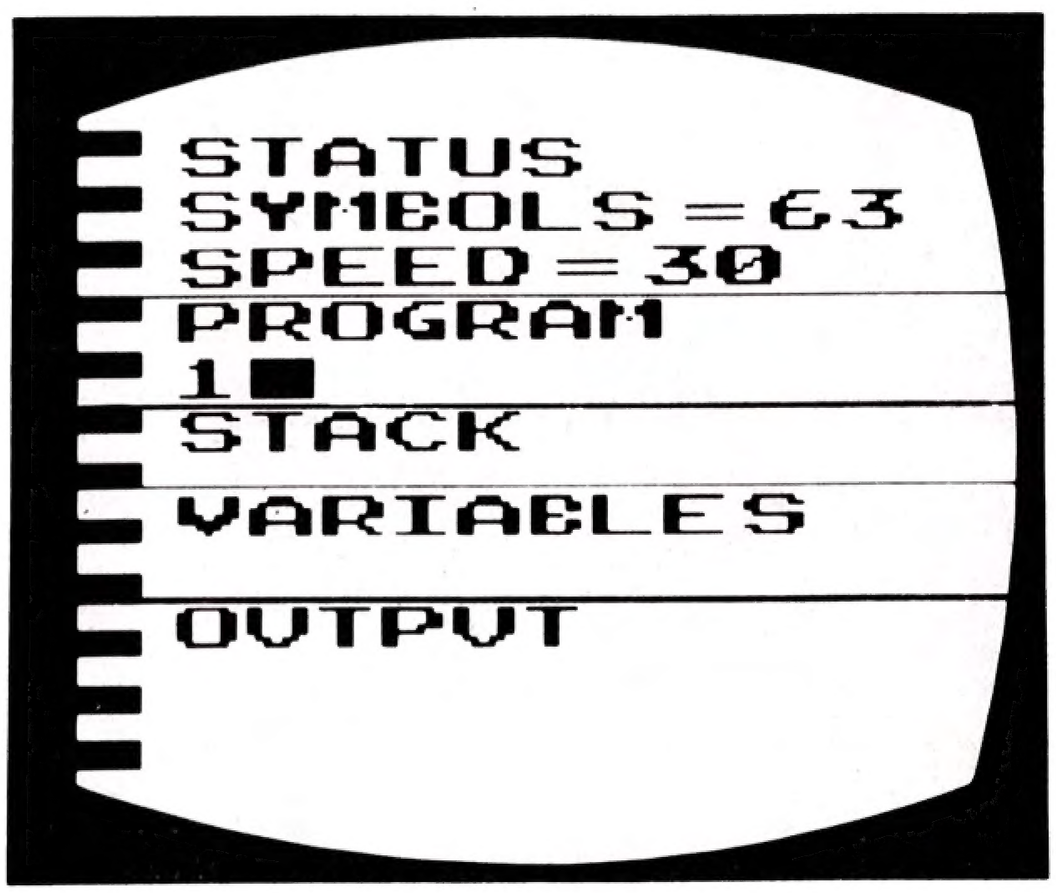
2. The **STACK** region will give you the temporary results of your program as the computer is running it.

3. The **VARIABLES** region will give you the value of each variable item in your program as it is being run.

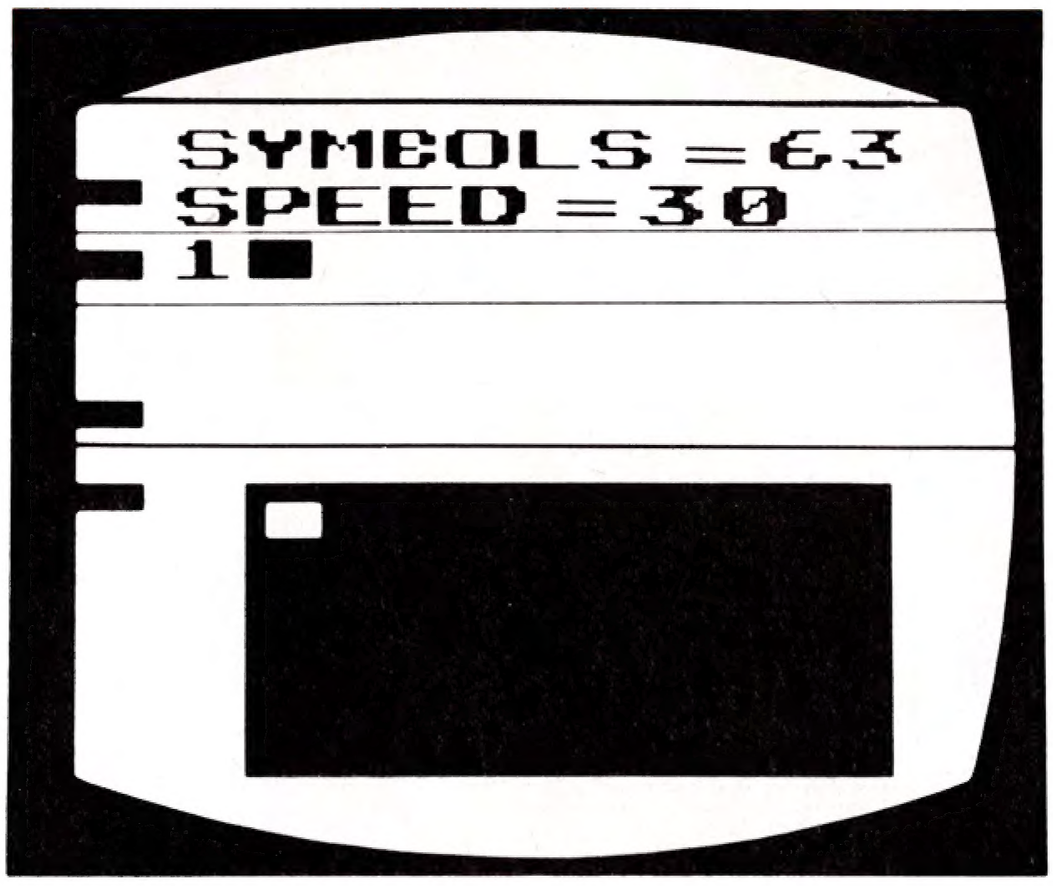
4. The **OUTPUT** region will show you the output being produced as your program is being run.

5. The **STATUS** region shows you the amount of memory available at any time for your program.

This region will also give you the speed of execution of your program.



6. The **GRAPHICS** region has two colored squares which can be moved around under your program control.



Before beginning your program, place the **left difficulty** switch in the **b** position. This will show you where each region is on your display. When the **left difficulty** switch is placed in the **a** position, the titles of each region will disappear from the display and the **GRAPHICS** region will come into view.

The **right difficulty** switch has no function in this Game Program.

THE CURSOR

Place the **left difficulty** switch into the **b** position and turn your console **off** and then **on** again. In the **PROGRAM** region there is a white rectangle. This is the cursor. Locate the *shift control key* at the center of the bottom row on the **LEFT CONTROLLER**. Push this key four times. The color of the cursor will change from white to red, from red to blue, from blue to green, and from green to white again.

The cursor is used to input your program. Each of the four colors on the shift key corresponds to the colored commands or inputs on your Keyboard. The white mode is used to give your computer commands, the other modes are used to insert symbols into your program.

MOVING THE CURSOR FROM REGION TO REGION

Make sure the **left difficulty** is in the **b** position and turn your console unit **off** and then **on** again. Push the **FORWARD** key and the cursor will move from **PROGRAM** to **STACK**. Push the **FORWARD** key again and the cursor will move from **STACK** to **VARIABLES**. Push the key again and the cursor will move from **VARIABLES** to **OUTPUT**.

By pushing the **BACKWARD** key you can move the cursor back to

the **PROGRAM** region. The **FORWARD** and **BACKWARD** keys can be pushed once for each region or held down.

Now place the **left difficulty** switch into the **a** position. The titles of each region will disappear and a portion of the **GRAPHICS** region will appear. Step the cursor through each of the regions again. Notice that the cursor will not move into the **GRAPHICS** region.

REMOVING THE REGIONS FROM THE DISPLAY

Place the **left difficulty** switch into the **b** position again and turn your console unit **off** and then **on**. On the left side of the Keyboard are a series of commands (white mode) that correspond with each of three regions: **STATUS, PROGRAM, STACK, VARIABLES, OUTPUT,** and **GRAPHICS**. Let's start with the **STATUS** region. Push the **STATUS** key once and the **STATUS** region will disappear from the screen and the **PROGRAM** region will move to the top of the display.

Push the **PROGRAM** key and the **PROGRAM** region will disappear and the **STACK** region will move to the top of the display. Push the **STACK** key and the **STACK** region will disappear and the **VARIABLES** regions will move to the top of the display. Push the **VARIABLES** key and the **VARIABLES** region will disappear causing the **OUTPUT** region to move to the top of the display.

Remove the **OUTPUT** region by pushing the **OUTPUT** key. This will also cause the **GRAPHICS** region to be fully visible on the display. Push the **GRAPHICS** key to remove the **GRAPHICS** region. Your display should show no regions.

Now push the **STACK** key. The **STACK** region will reappear on your display. Remove the **STACK** region, and bring up the **OUTPUT** and **VARIABLES** regions. You may display or remove any region with the **left difficulty** switch in either the **a** or **b** position.

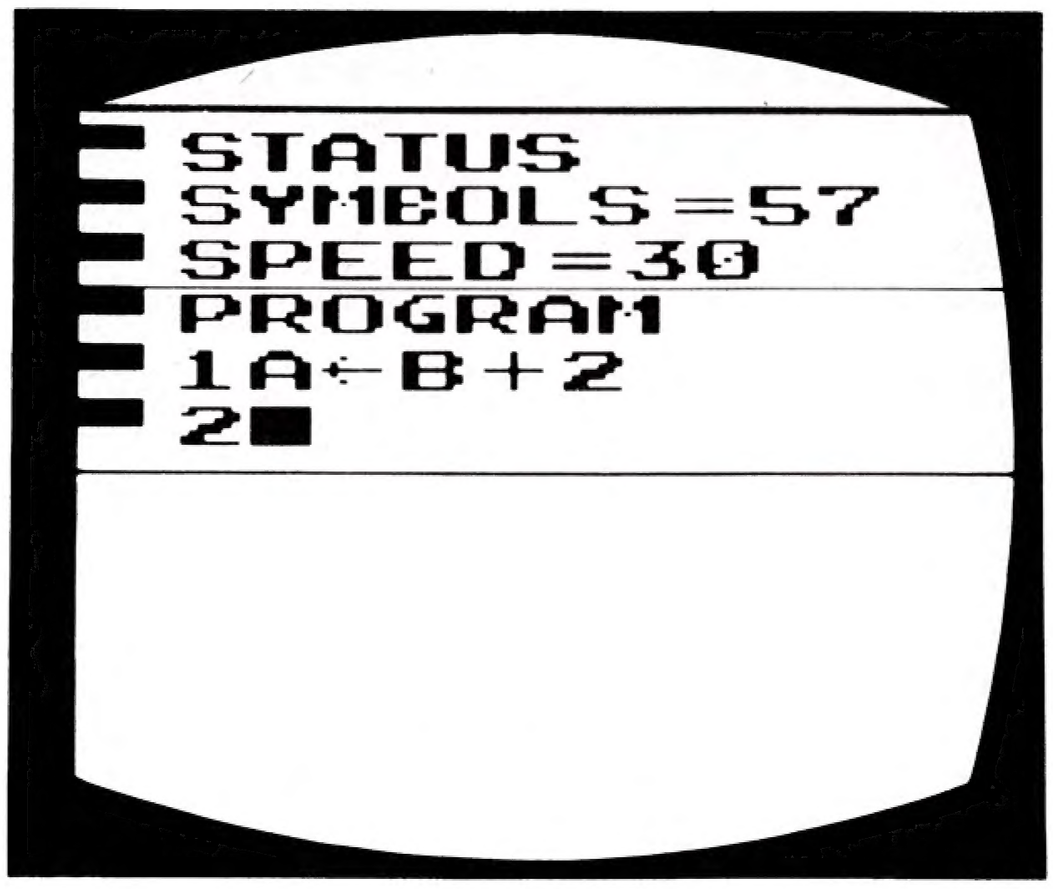
It is important to note that whether or not the various regions are displayed as no effect on the execution of a program.

RUNNING A PROGRAM

Let's do a simple program. Make sure the **left difficulty** switch is in the **b** position and turn your console unit **off** and then **on** again. (Another way to erase a program and reset all values is to depress the **game select** switch. Pushing the **game reset** switch erases all values and returns a program to its beginning without erasing the program.)

Remove the **STACK, VARIABLES, OUTPUT,** and **GRAPHICS** regions from the display. The cursor will be in the white mode and to the right of the number **1** in the **PROGRAM** region. Change the cursor to the blue mode and push the **A** key. The letter **A** will appear on the display next to the **1**. (Each line is numbered, making it possible to see where one line ends and the next one begins.) Now change the cursor to the red mode and push the ← key. A small arrow will appear next to the **A**. Now go back to blue and input **B**. Changing the cursor to red, input **+** and the number **2**. Now go back to the white mode and input **New Line**. You must always be in the white mode in order to start a new line in your program.

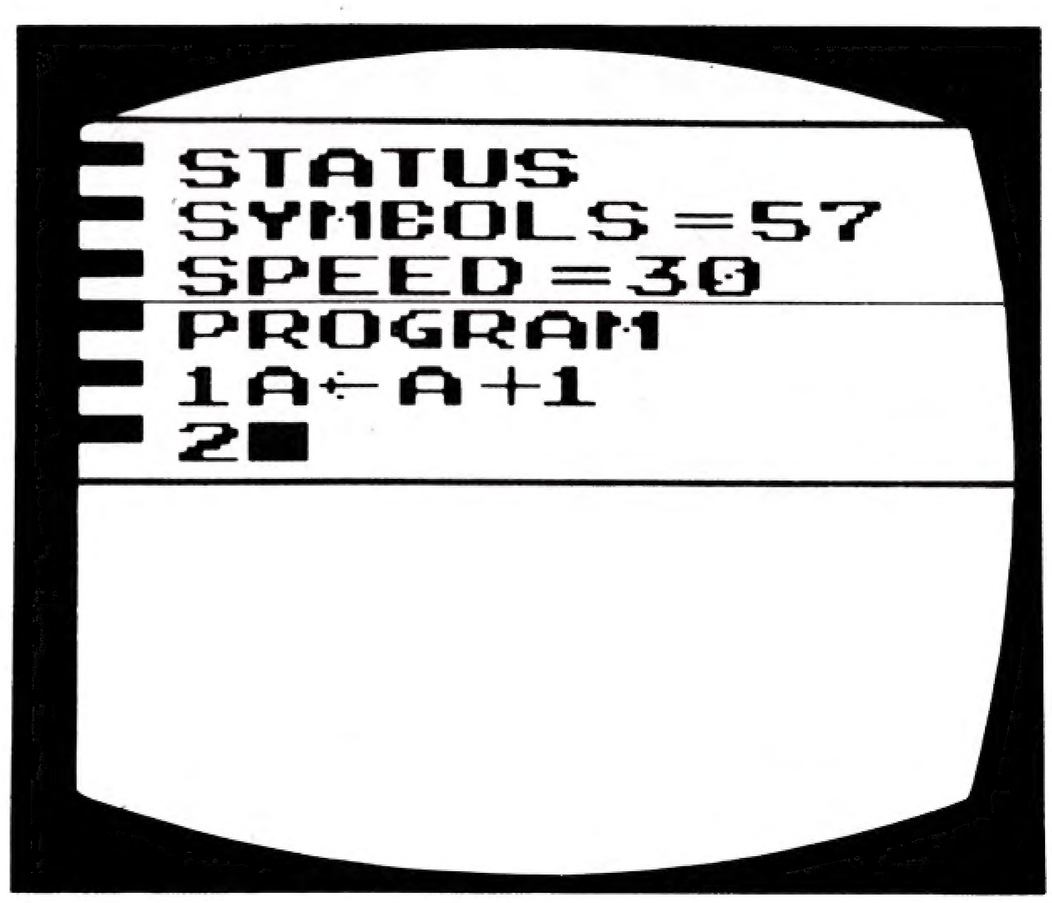
Your display should look like this:



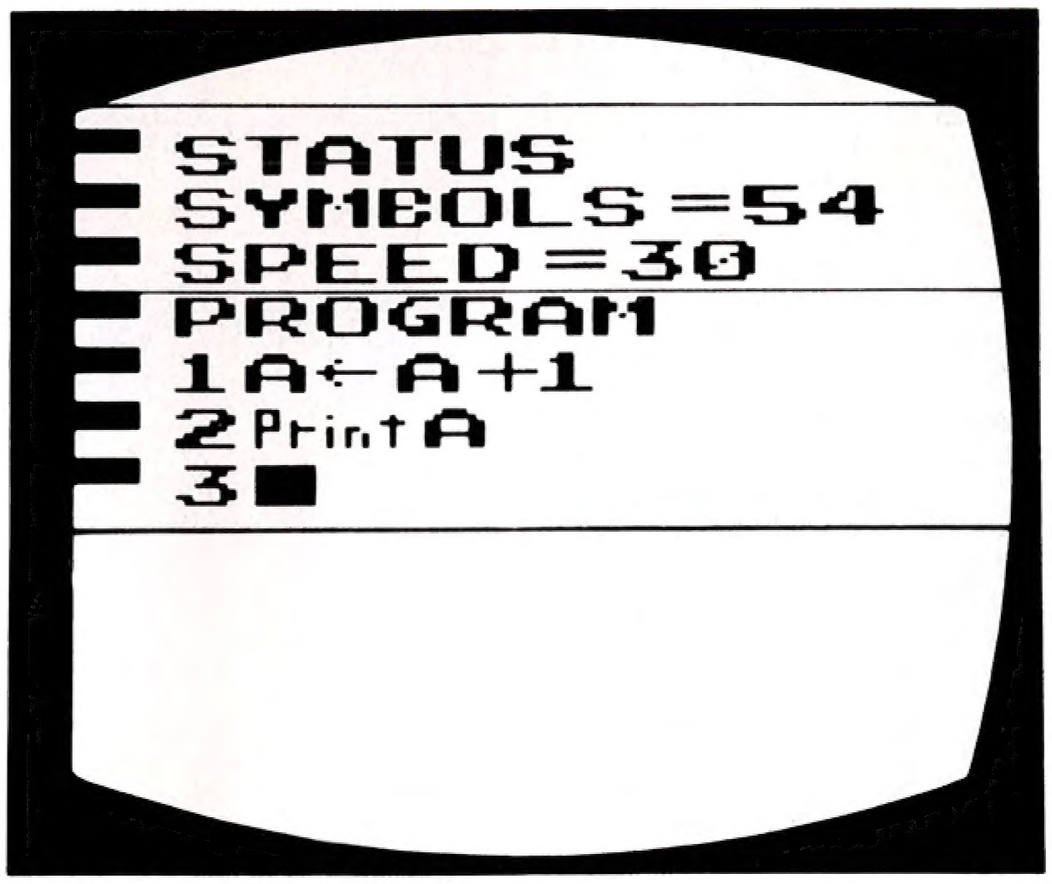
If you make an error in entering your program, it can be erased using the **ERASE** key. Notice that the **ERASE** key is **not** color coded. It can be used when the shift key is in any color mode.

Using the line that you have just entered into your program, let's do an exercise. Push the **ERASE** key once. The cursor will move from line **2** directly to the right of the **2** on line **1**. Push the **ERASE** key again and the **2** will be erased from the program. Now enter **1**, using the red mode. Change the cursor to the white mode and using the **BACKWARD** key, move the cursor until it is directly to the right of the **B** in the program. Push the **ERASE** key and the letter will be removed from the program. Now replace it with **A** (blue mode). In the white mode, use the **FORWARD** key to move to the end of line **1** and input **New Line**.

Remember that the cursor should not be on the symbol to be erased but directly to the right of it. Your screen display will now look like this:

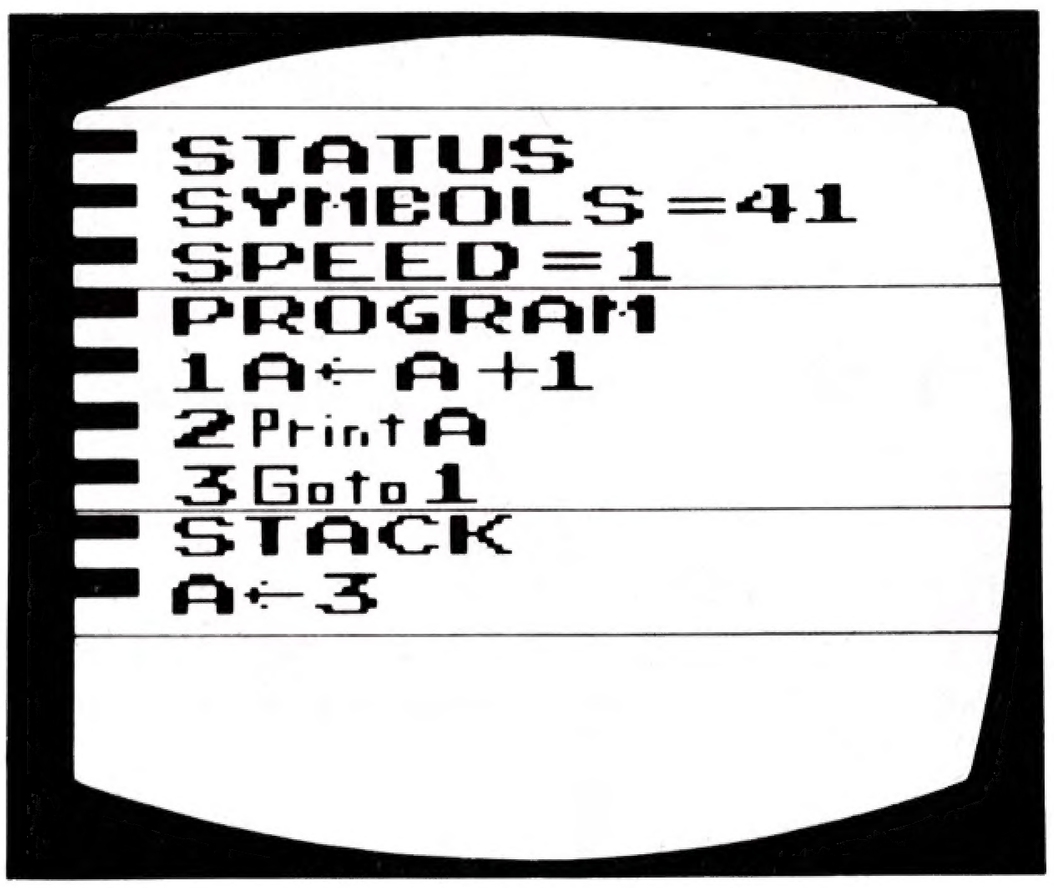


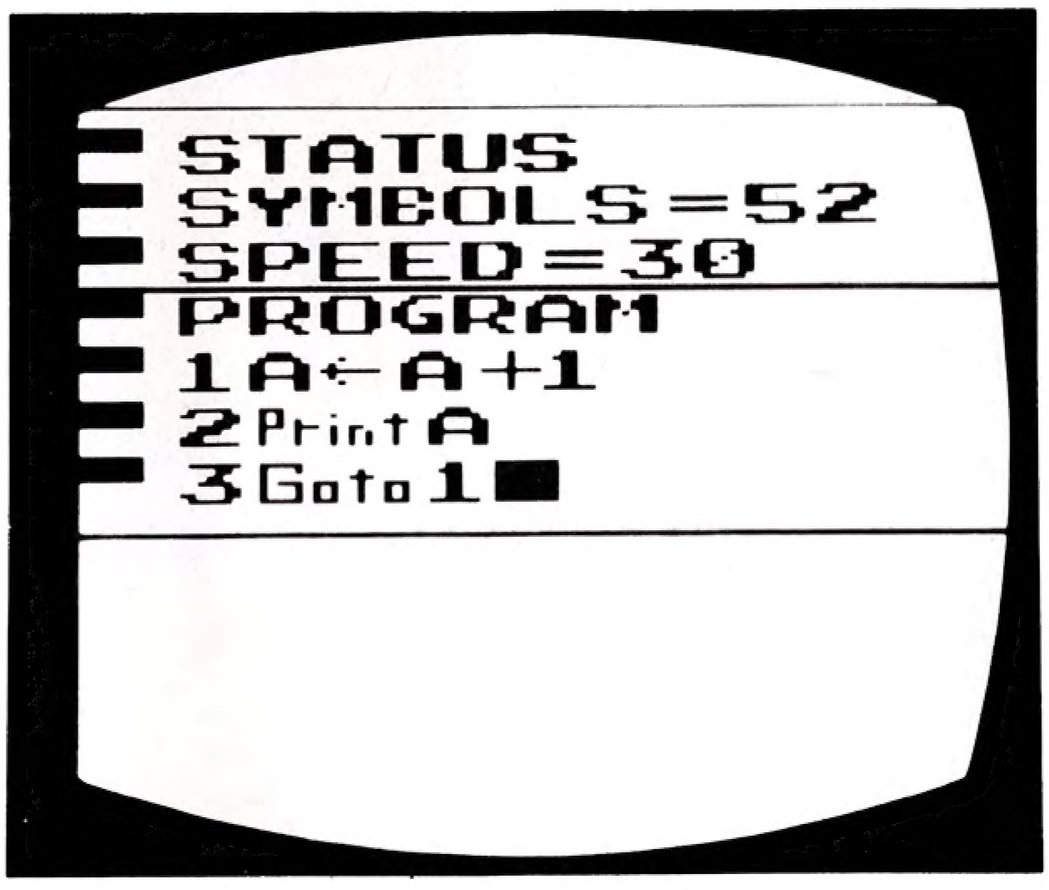
The cursor is directly to the right of the **2** in line **2**. Now input **PRINT** using the green mode. Then, with the blue mode, input **A** and go to a New Line. Your display will look like this:



Before we start the program, input **SPEED = 1**. Now depress the **STACK** key. Press the **RUN/HALT** key twice and the program will begin. You can watch the computer work each part of the program.

Now with the cursor in the green mode, input **GOTO** and then, in the red mode, input **1**. Before we do anything else, let's look at the display again.





To stop the program press **RUN/HALT**. As noted before, pressing the **game reset** switch erases all values in your program (without erasing the program itself), and returns the program to its beginning.

Let's analyze the program step by step as the computer runs through it. Change **SPEED** to **60**. With the cursor in the white mode depress the **STEP** key. This will take each part of the program one step at a time, each time it is pressed.

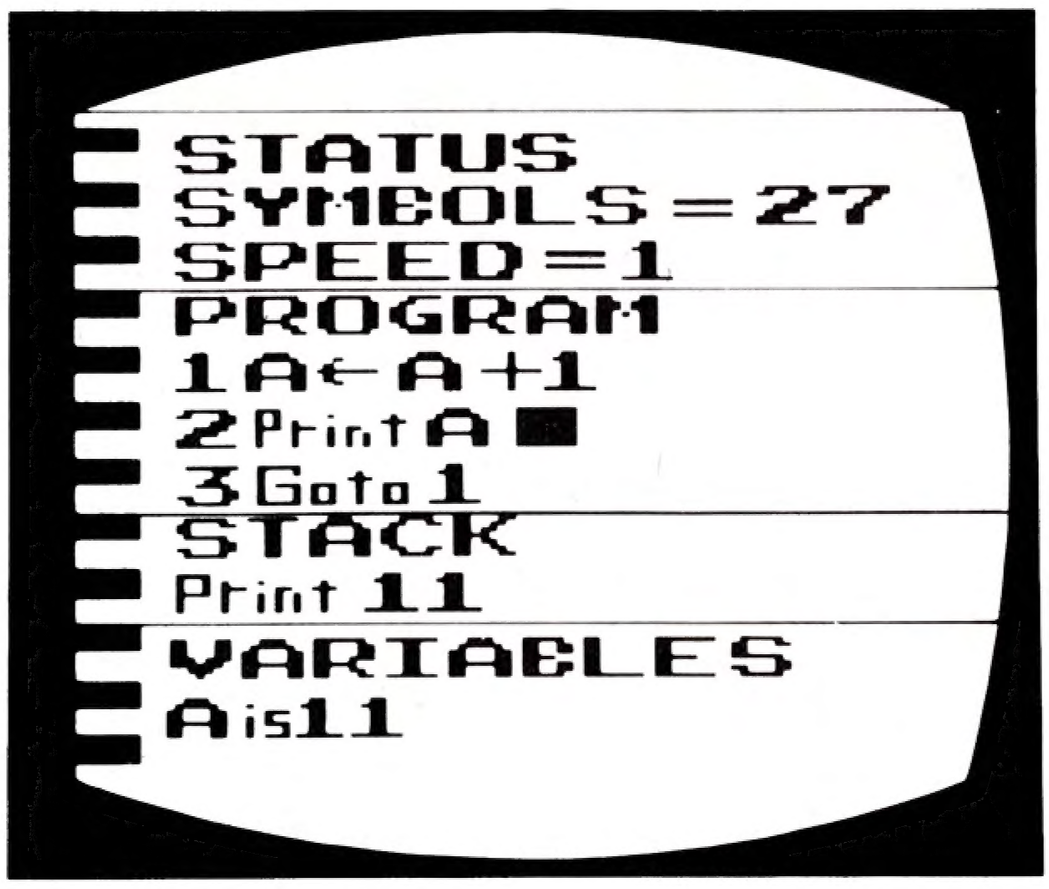
Notice that the **STATUS** region shows we have 52 "bytes" or symbols of memory remaining. The **SPEED** is set at 30 **(SPEED = 30)**. Put the cursor into the white mode and depress the **SLOWER** key. Notice that every time you depress this key the speed decreases. Depress the **FASTER** key and the **SPEED** will increase.

In line **1** the program reads **A ← A + 1**. The computer reads this as **A** "becomes" **A + 1**. Watch the **STACK** region as you step through line **1**. In line **2**, you have told the computer to **Print A**. This means you want the computer to print the value of A (as determined by line

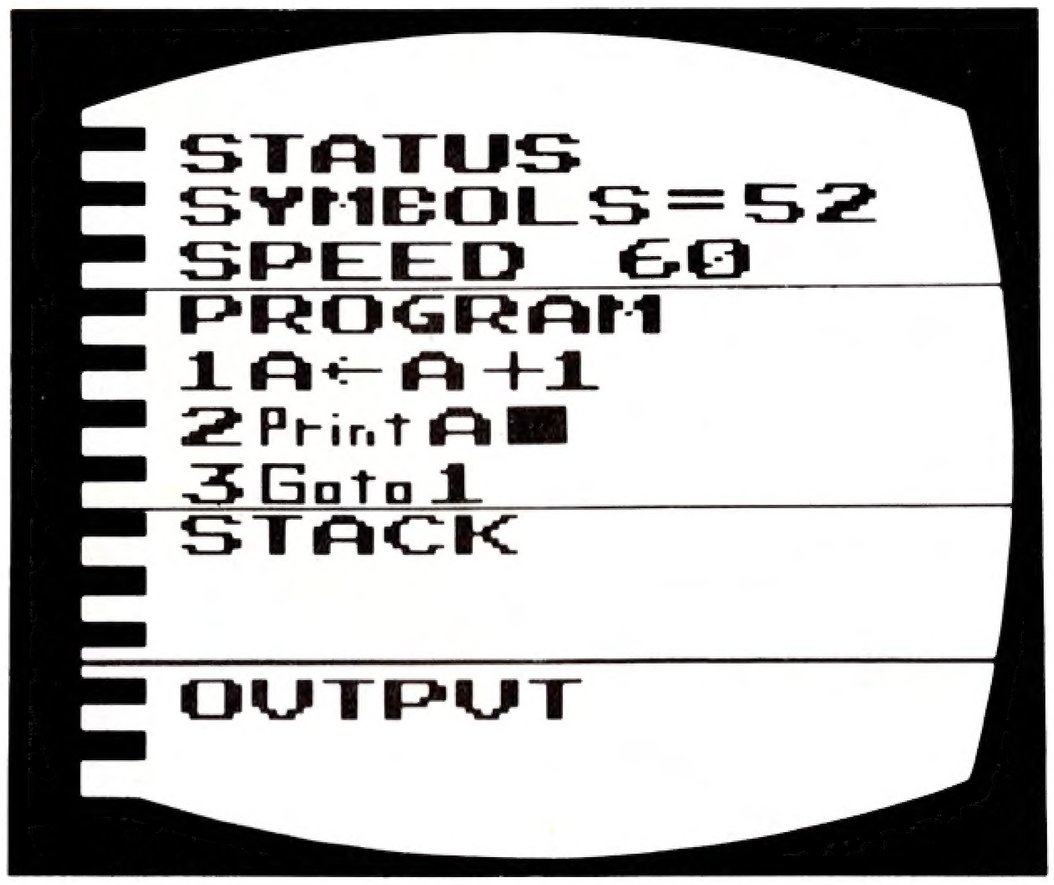
1) in the **OUTPUT** region.

We'll see how that works later. Line 3 tells the computer to **Goto 1**. The computer is to return to line 1 and find a new value for **A**. Each time the program is run through it will find a new value for **A**, print that value, and then return to line 1. The computer will continue to run through the program in this manner until all the "memory" is used. The amount of remaining memory is shown in the **SYMBOLS** area of the **STATUS** region.

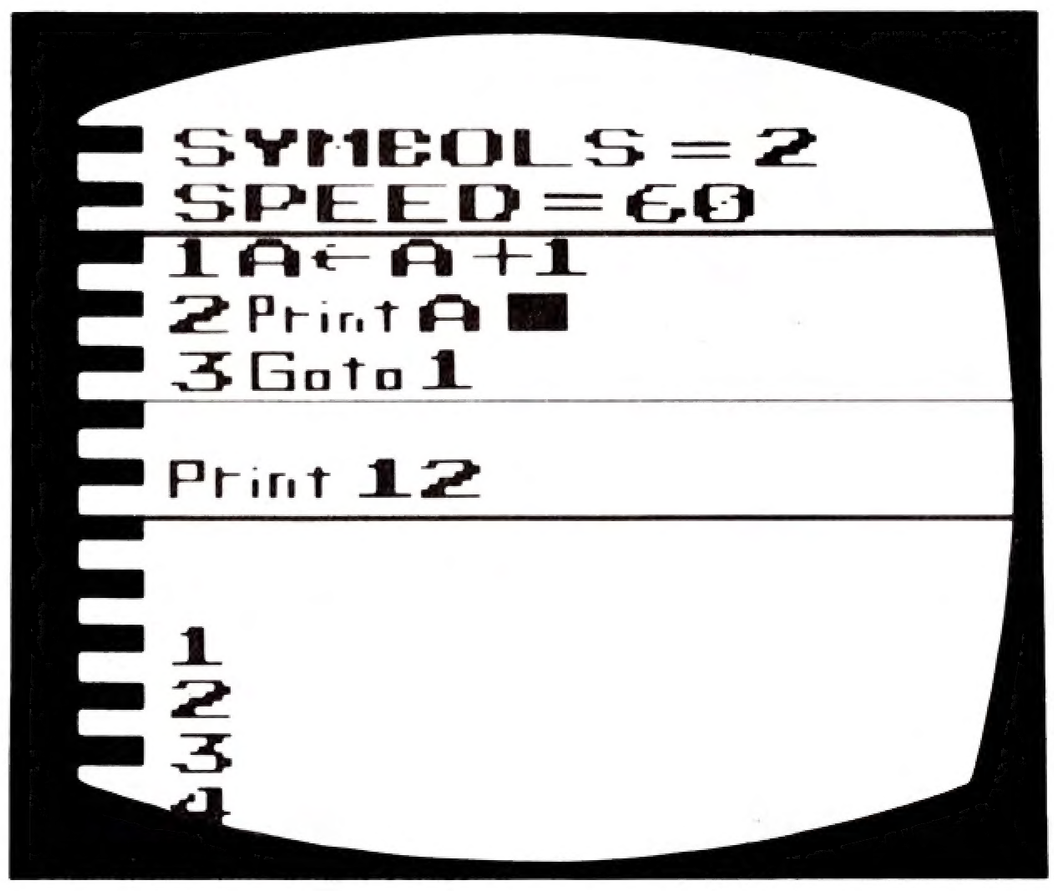
Bring up the **VARIABLES** region on the display. Change the **SPEED** to 1 and clear the values of the program by pressing the **game reset** switch. Push the **RUN/HALT** key and watch as the computer runs through the program. The computer will show you the current value of **A** at each step of your program. Your screen display will look like this:



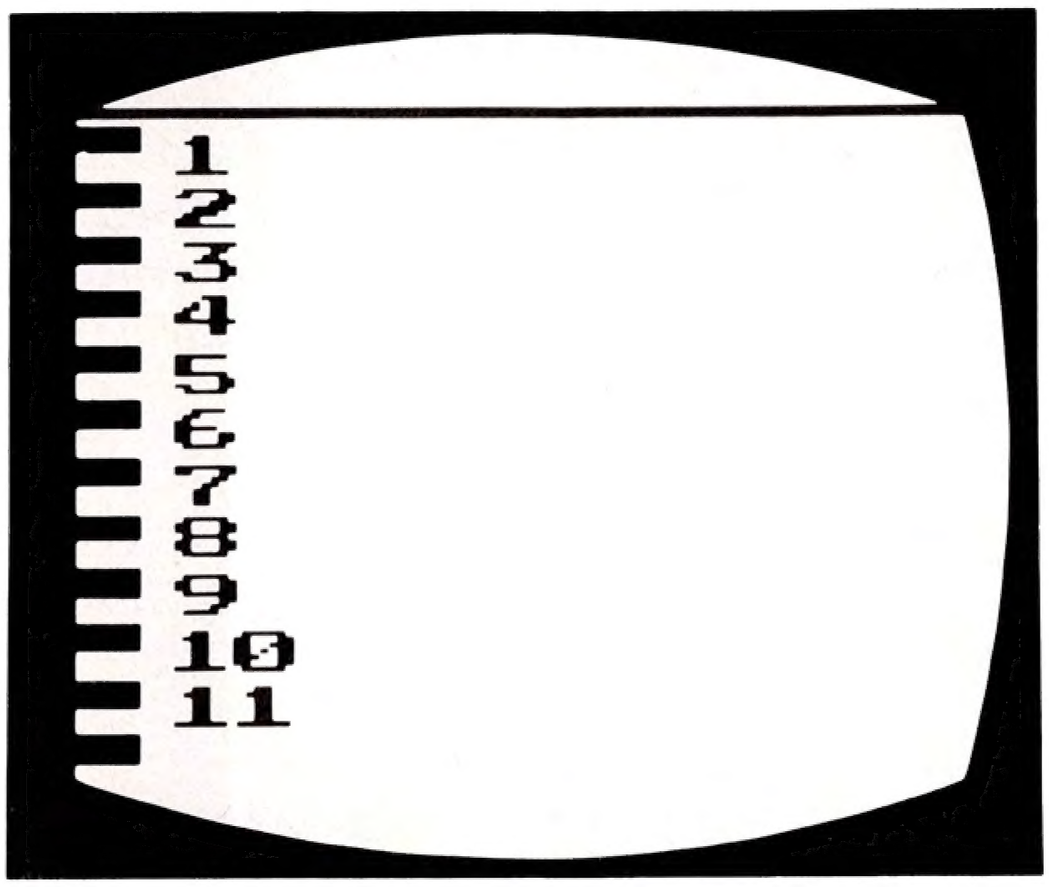
Stop the program and clear the values (**game reset**). Remove the **VARIABLES** region and bring up the **OUTPUT** region on the display. Change **SPEED** to 60. The display will now look like this:



Press the **RUN/HALT** key and start the program. In line **2**, the computer is instructed to **PRINT A**. When it reaches this command, it will print the current value of **A** in the **OUTPUT** region. Watch the **SYMBOLS** area of the **STATUS** region. Even though 1 shows in the **OUTPUT** region on your display, the computer is still printing the changing value of **A**. Put the **left difficulty** switch in the **a** position. Your display will now look something like this:

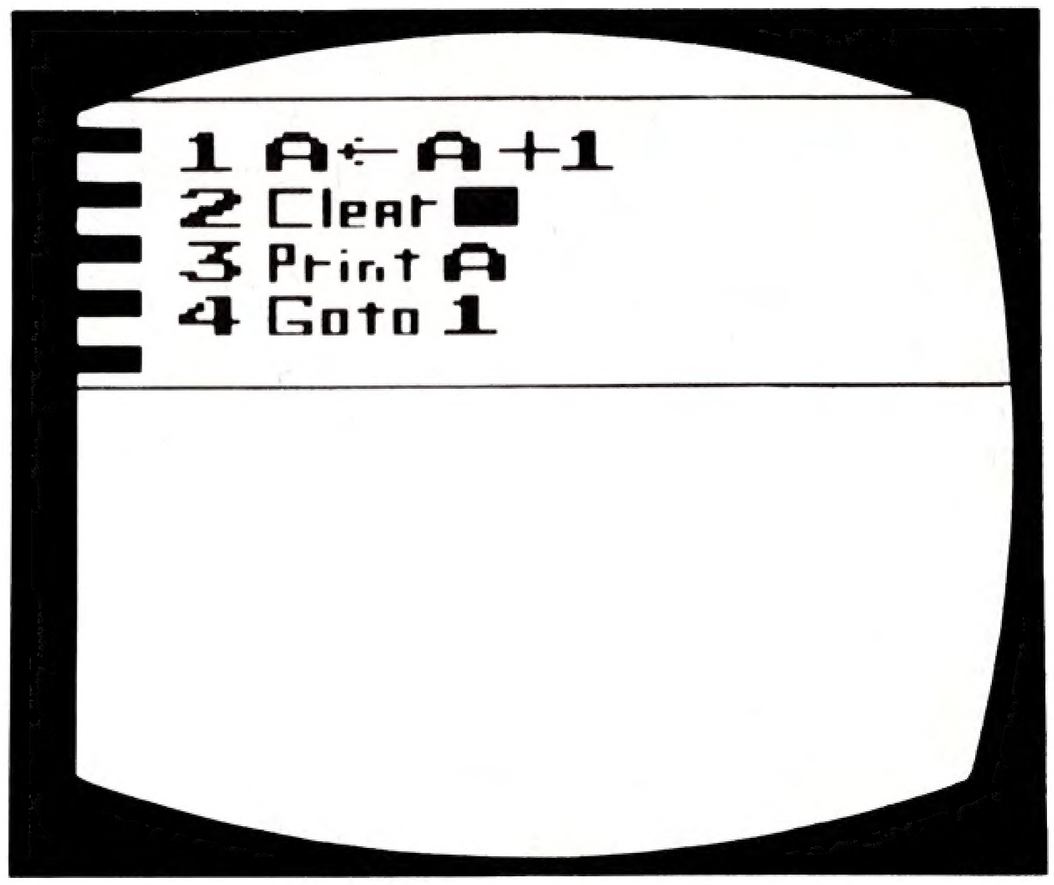


How can we show more of the **OUTPUT** region on the display? Remove the **STATUS, PROGRAM,** and **STACK** regions from the display. Your display will look like this:



Suppose we don't want to overload the **OUTPUT** region of the program? Stop the program and erase the values with the **game reset** switch. Remove the **OUTPUT** region from the display and bring up the **PROGRAM** region. Using the **FORWARD** key, move the cursor to the end of line 1, so that your display looks like this:

Now depress the **NEW LINE** key. We will insert a new command here. Input **CLEAR** (green mode). Your display will look like this:



Bring up **STACK, VARIABLES,** and **OUTPUT**. Leave the **left difficulty** switch in the **a** position and start the program. As the program passes line **2 CLEAR** it erases the value of **A** in the **OUTPUT** region and then prints the next value of **A** in line **3**.

BASIC PROGRAMMING can only work with two-digit numbers, so as it reaches 99 it will "wraparound" and begin again showing A's value to be 0.

USING THE NOTE FUNCTION

Each time the program stores a number into **NOTE** (red mode), a note from the musical scale will be sounded by your television speaker.

Let's do some programs to demonstrate. If you have any programs in your Video Computer System, depress the **game select** switch on the console unit.

Input the following:

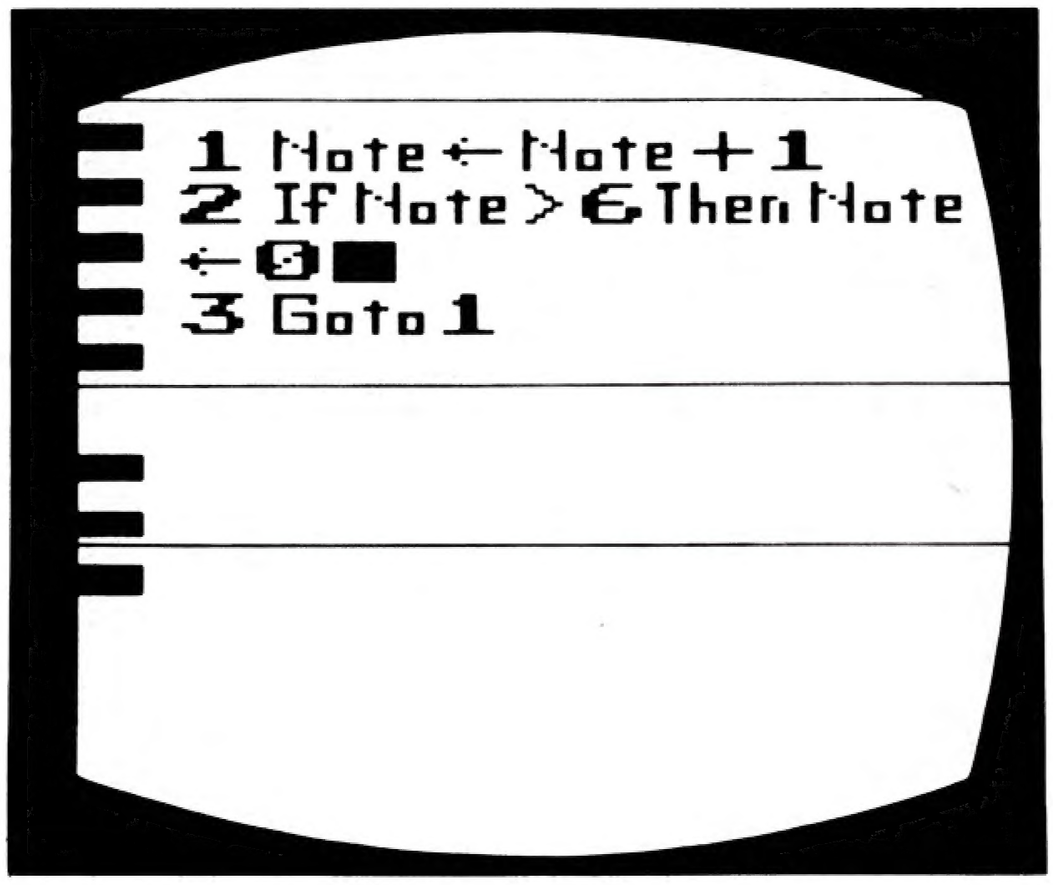
```
1  Note ← Note + 1
2  Goto 1
```

Now start the program. Slow the program down and watch it being worked in the **STACK** region. Also notice that the program will print the current value of **NOTE** in the **VARIABLES** region.

Stop the program and insert a new line **2** (in the white mode, move the cursor to the end of line 1 and press **NEW LINE**. The previous line **2** will become line **3**).

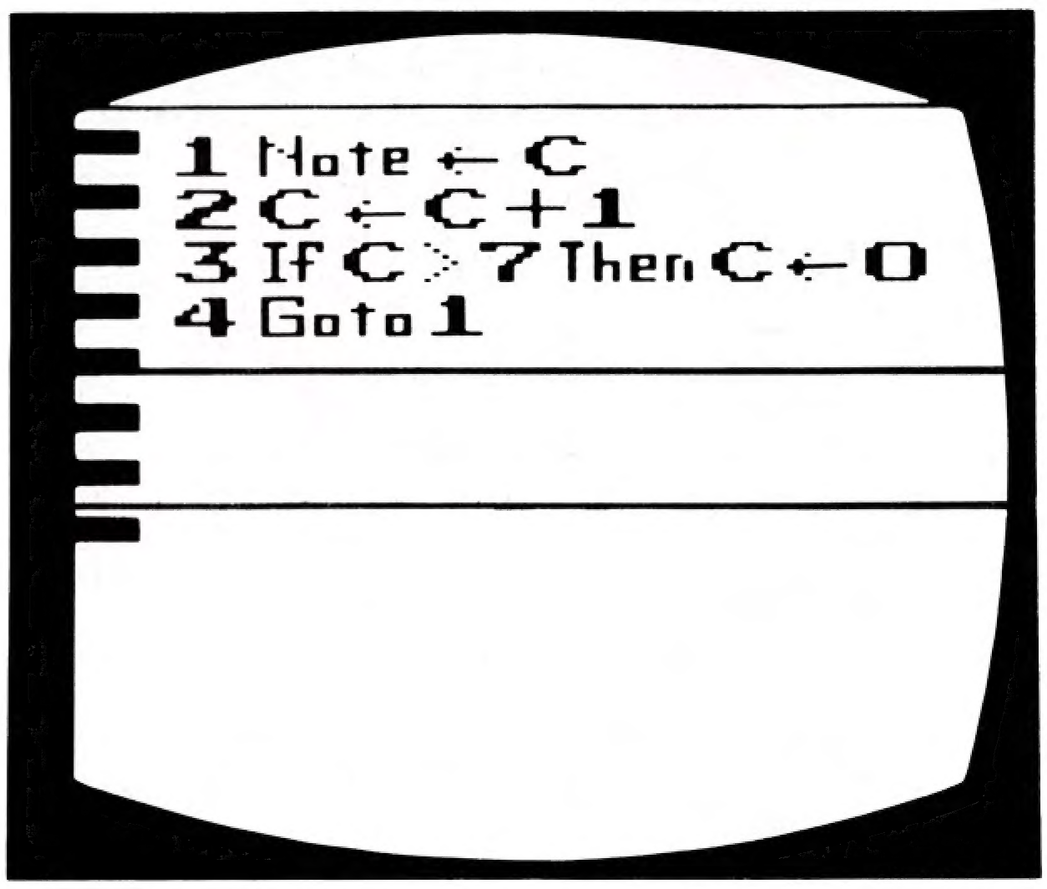
```
2  If Note > 6 Then Note ← 0
```

Your display will look like this: (If **STACK** and **VARIABLES** regions are removed.)



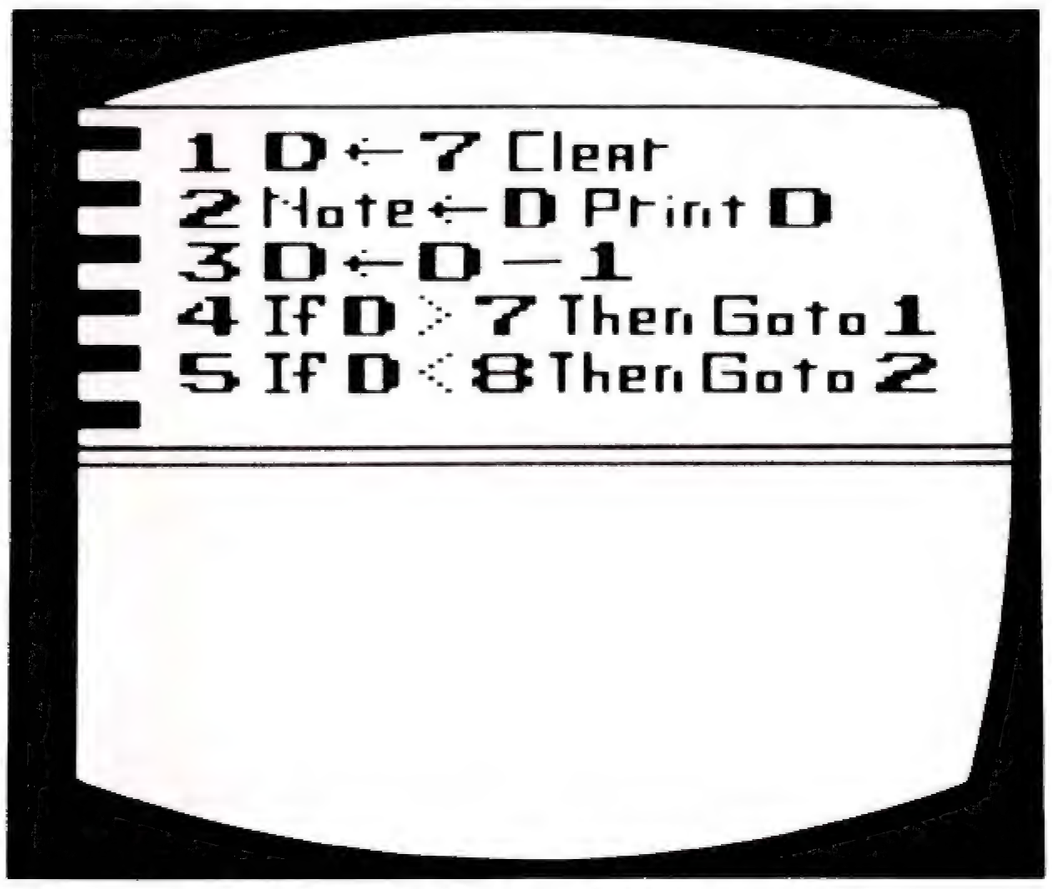
With the **IF** and **THEN** commands, we are instructing the program that IF something happens THEN it is to do something else. In this case, **IF** Note is more than 6, **THEN** Note is to be changed to 0. Start the program and watch it being worked in the **STACK** region. Notice that as the value of Note reaches 7, it becomes more than (>) 6 and the program changes the value to 0.

Let's do another program to achieve the same result in a different way. Input the following:



Run the program. Notice that the same results are achieved, except that the notes are spaced evenly. If you want the program to give you the value of **C** in the **OUTPUT** region, input **Print C** and **Clear** as instructions anywhere in the program. To cut down on the flicker in the **OUTPUT** region as the program is giving you the value, insert a comma, (green mode) after the value in the Print line, **Print C.**

You may use any letter of the alphabet for the variable in your programs. The following program has all the key functions we have learned so far and uses nearly all the available "memory" in the computer. After entering this program, remove the program from the display, bring up the **STATUS, STACK,** and **OUTPUT** regions, and run the program.



In this program we have given two **IF/THEN** commands. If the conditions set up in the first command are not met (line 4) the computer will move to the next line (line 5). Given enough memory, there can be several commands between any two **IF/THEN** commands.

In some cases we can put two commands on one line, as in line 1 and line 2 in the above program. In this way we can save some of the memory for later in the program.

USING THE KEY AND PRINT FUNCTIONS

The **KEY** function (red mode) is used to input a variable while the program is running. The program will evaluate **KEY** and replace it with a number that you input from the right side of the Keyboard. If no number is inputted, then the program will read **KEY** as **0**.

Here's a simple program using the **KEY** function:

Start the program and push some of the keys on the right side of the Keyboard. With practice you may be able to play a tune.

The **KEY** function can also be used for programming in the **GRAPHICS** region as we will see later.

Try this program using the **KEY**, **NOTE**, and **PRINT** functions:

Watch this program in the **OUTPUT** region. Notice that as you input a number from the right side of the Keyboard the program will play that tone and display that number in the **OUTPUT** region.

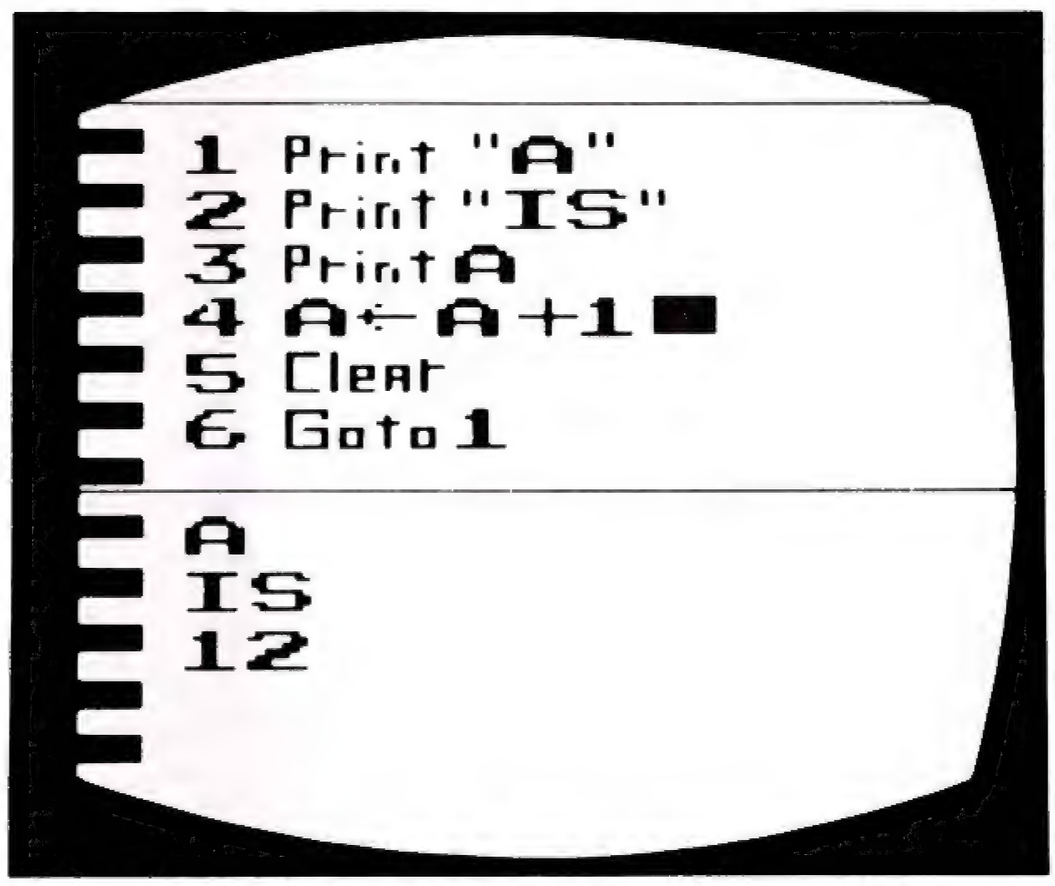
Now let's make a minor modification in the program. In line **2**, insert a comma (,) after **Print A** and start the program. By inserting the comma here you have told the program that you want as many of the variables printed on one line as possible.

USING THE PRINT FUNCTION

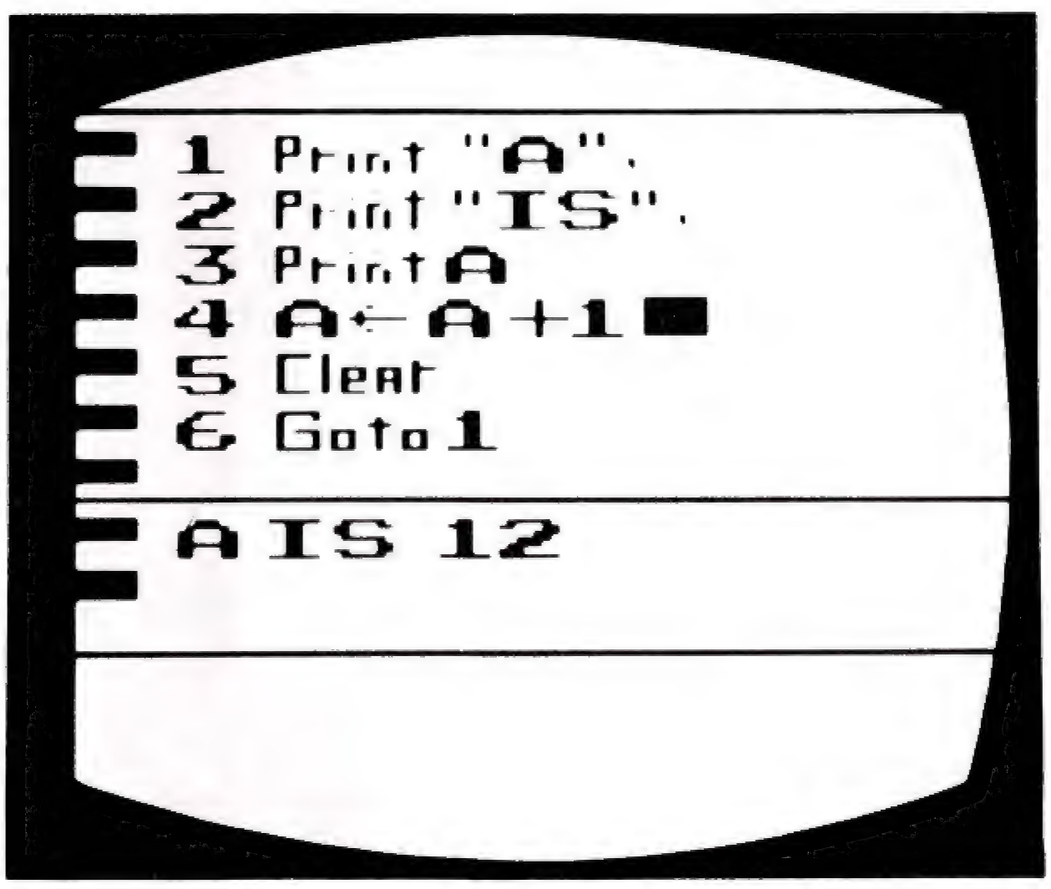
As we showed in earlier programs, you can use the **PRINT** function to instruct the program to print the value of a variable in the **OUTPUT** region. The **PRINT** function can also be used to instruct the program to print words on the display. Words to be displayed must be enclosed in quotation marks ("").

Set the **SPEED** at **8** and input the following program:

Now run the program and watch the **OUTPUT** region. The program prints the words you have inputted, but they are "stacked."

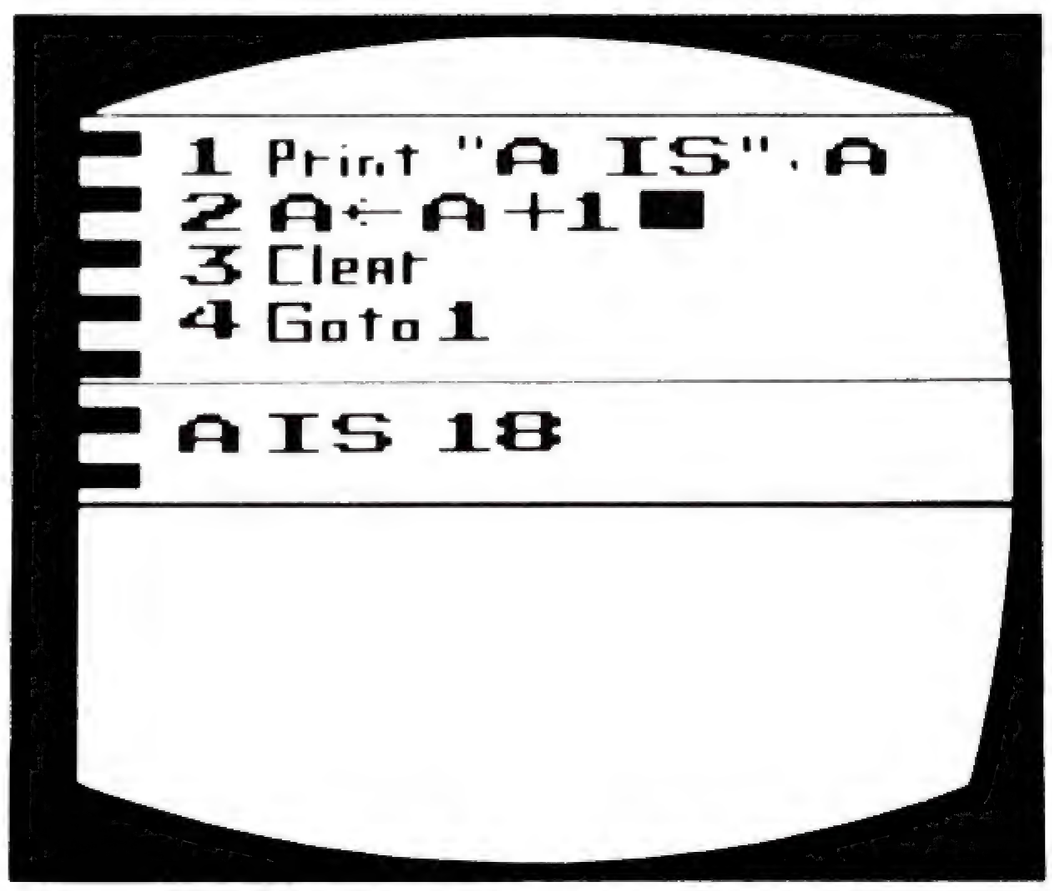


Let's change the program slightly by inputting a comma (,) at the end of line **1** and line **2**. The program now looks like this:



The comma (,) in line **1** has instructed the program that whatever is on line **2** is to be displayed in the **OUTPUT** region on the same line as the instruction on line **1**. The comma in line **2** has instructed the program that the line **3** instruction is to follow line **2**. These instructions can be shortened by changing the program to read:

The program can be further shortened:



The comma (,) in line **1** has instructed the program that whatever is on line **2** is to be

Writing the program in this fashion will also save some memory.

USING THE GRAPHICS REGION

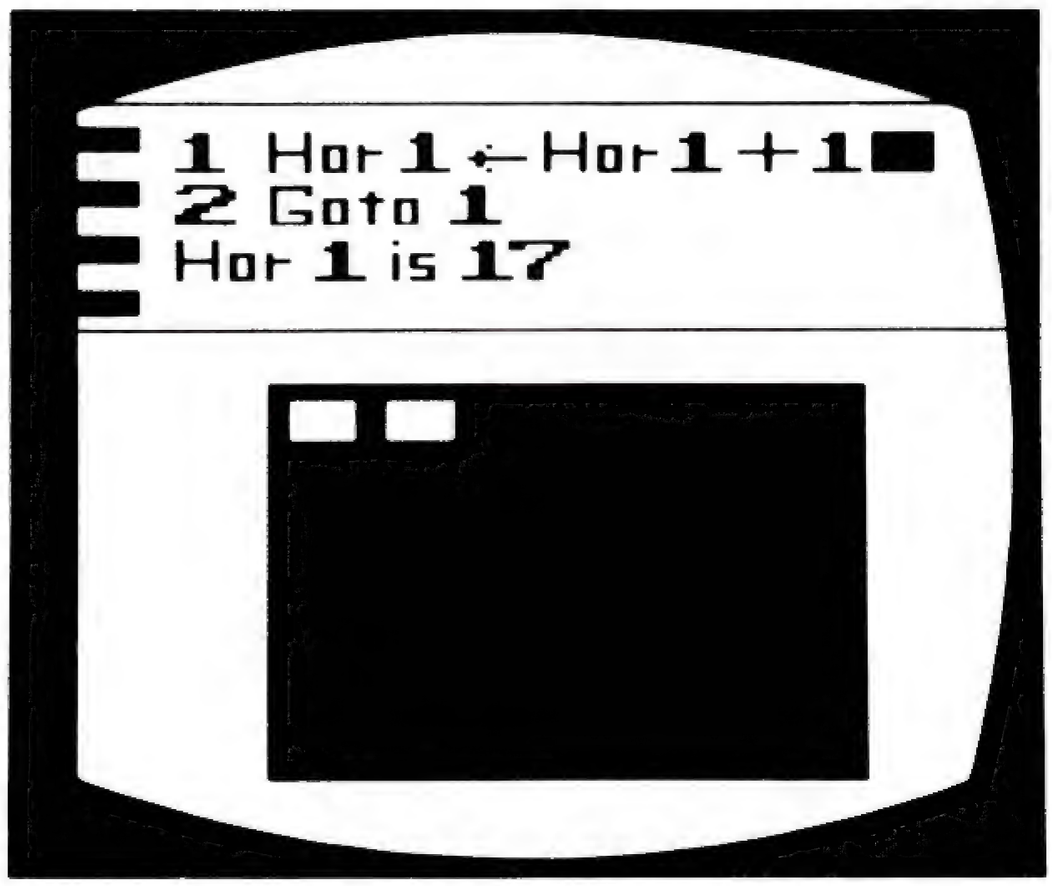
The **GRAPHICS** region is the blue rectangular field on your television screen. At first glance it appears to contain one red square in the upper left corner. The red square actually covers a white square and both may be moved independently on the field under your program control.

To move the squares you must change their coordinates. The red square is object number 1. Its horizontal coordinate is represented on the Keyboard by **Hor 1** and its vertical coordinate by **Ver 1. Hor 2** and **Ver 2** are coordinates of object number 2, the white square.

Both objects or squares start out in the upper left corner of the blue field, with only the red square showing. The upper left corner is the zero (0) position or origin. When the variables **Hor 1, Ver 1, Hor 2,** and **Ver 2** are not defined (given values) they have values of 0 (therefore causing the squares to remain in the upper left corner position).

When a coordinate is assigned a value other than 0, (e.g., **Hor 1 ← 10**), the object involved will jump to the corresponding position on the blue field when the program is executed.

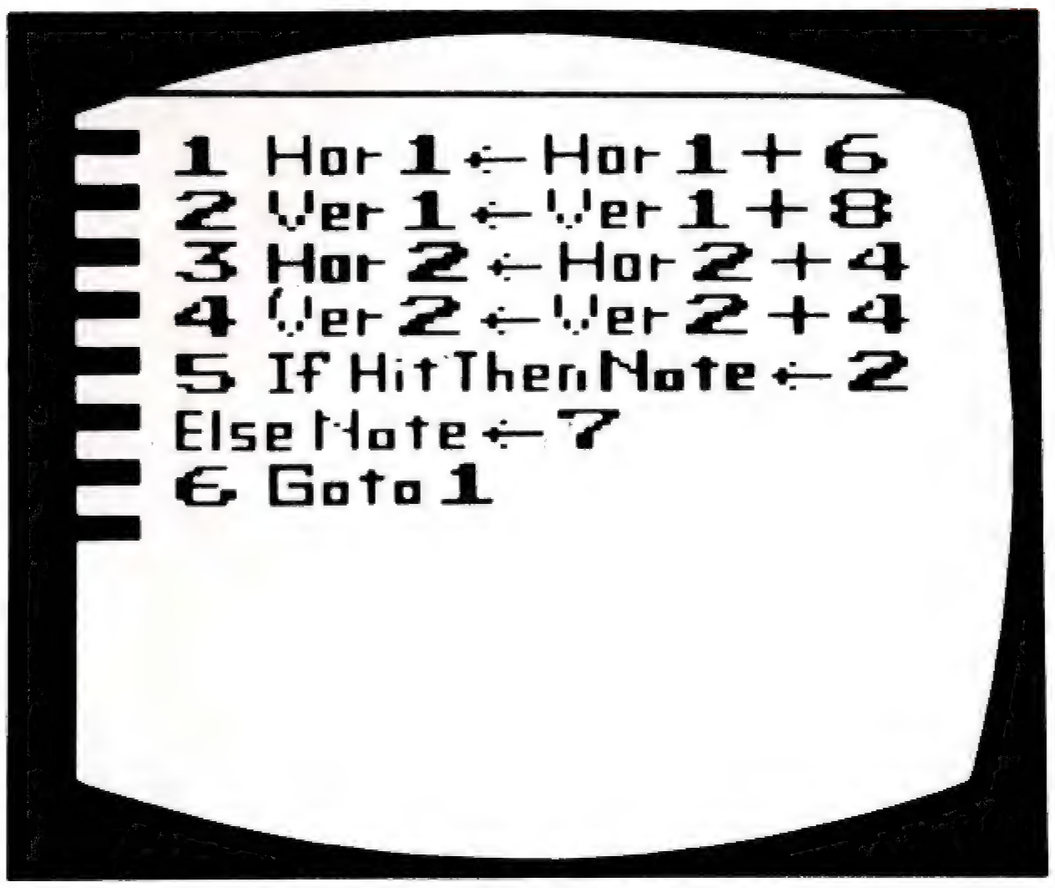
Input the following program:



This program will move the red square to the right, one horizontal space at a time. Run this program and you will see the red square moving slowly to the right. Check the **VARIABLES** region and you will see the value of **Hor 1** increasing. It will increase until it reaches **99**, the largest number allowed, and then drop back to **0**. When **Hor 1** reaches **99**, the red square will have reached the far right side of the blue field. At this point, the red square "wraps around" and starts again from the far left side of the field, with **Hor 1** at **0**.

The horizontal coordinates **(Hor 1** and **Hor 2)** then, range from 0 to 99. The vertical coordinates **(Ver 1** and **Ver 2)** also range from 0 to 99, with 0 being the position at the top of the blue field, and 99 being the position at the bottom.

Input the following program:



Remove the program from the screen and make sure the **GRAPHICS** region is fully visible. Set the speed at **60** and run the program. This program shows you one way to move the squares on the field. It also reveals how the **HIT** and **ELSE** functions may be put to use.

In line **5** the program is instructing the computer to sound the **2** note **IF** the squares **HIT**, which they do periodically. To **HIT** means the squares must occupy the same coordinates. Otherwise (**ELSE**) the computer is instructed to play the **7** note, which it will do until the squares hit.

(Remove **ELSE NOTE ← 7** from line **5** and note **2** will be played when the squares hit and no other note will be sounded.)

USING THE MOD FUNCTION

Mod is an arithmetic operator, much like the division operator (÷). **BASIC PROGRAMMING** employs integer division, which means it works with whole numbers only, and leaves no remainder. For example, what is 14 ÷ 5? You might say 2 and 4/5, or 2, remainder 4. In **BASIC PROGRAMMING** 14 ÷ 5 is 2. Although 5 divides into 14 twice (making 10), with 4 left over, **BASIC** uses whole numbers only, so the answer it gives you is **2**.

The computer will give you the same answer for 12 ÷ 4 as it will for 13 ÷ 4. the answer in both instances is 3, since 4 will divide into both 12 and 13, 3 whole times. The computer will not recognize the remainder of 1 in the case of 13 ÷ 4.

Now for the **Mod** function. **Mod** gets the remainder left over after dividing the first number into the second number. So, **14 Mod 5** is **4.** Five divides into 14 twice (making 10) with 4 left over. **Mod** gets the remainder left over, therefore **Mod** is **4**.

What is **13 Mod 4?** The answer is **1** since **1** is left over after dividing **4** into **13** (which makes **12**). How about **12 Mod 4**? In this case, **Mod** is **0** since 4 divides into 12 evenly with nothing or 0 remaining.

Normally dividing by 0 is undefined, mathematically speaking. In

BASIC PROGRAMMING, dividing by 0 just gives a result of 0, so that 5 ÷ 0 is 0.

OPERATOR PRIORITIES

In an equation, the arithmetic operators (+, —, x, ÷, Mod, etc.) are worked out in order of an established priority. **BASIC PROGRAMMING** uses the following operator priority guide:

1. x ÷ (Highest)
2. + —
3. Mod
4. =
5. ← (Lowest)

USING PARENTHESIS

Using parenthesis (green mode, left side of Keyboard), gives priority to whichever numbers the parenthesis surround when working out an equation.
For example, when working out the following equation:

A 5 + 3 x 2 Mod 7

the first step is 3 x 2 = 6
the second step is: 5 + 6 = 11
the third step is: 11 Mod 7 = 4
A = 4

However, the same equation with parenthesis inserted, is worked out differently and gives a different answer for A.

A (5 + 3) x 2 Mod 7

first step: 5 + 3 = 8
second step: 8 x 2 = 16
third step: 16 Mod 7 = 2
A = 2

SAMPLE PROGRAMS

Here are some sample programs. They will help you see the wide range of possibilities you have to work with. Remember to pay close attention to how a particular command (**IF, THEN, HIT, ELSE, PRINT, KEY**, etc.) affects a program. Remember too that a **KEY** command may depend on your input from the right side of the Keyboard for successful operation of the program.

Whenever a particular program confuses you, stop the program, reset it to the beginning, and **STEP** through it. By watching the program in the **STACK** region and seeing how the computer works out each step, you should be able to understand what is happening and why it is happening.

After running the programs and becoming familiar with the fundamentals of computer programming in general, you'll be well on your way toward writing some programs of your own.

HIT

```
1 Clear PrintHit
2 Hor 2←Hor 2+1 Mod
3 3
3 Goto 1
```

MUSIC

```
1 Note←If Note>4
Then Note—11 Else
Note+1
2 Note←Note+2
3 Note←Note—2
4 Goto 1
```

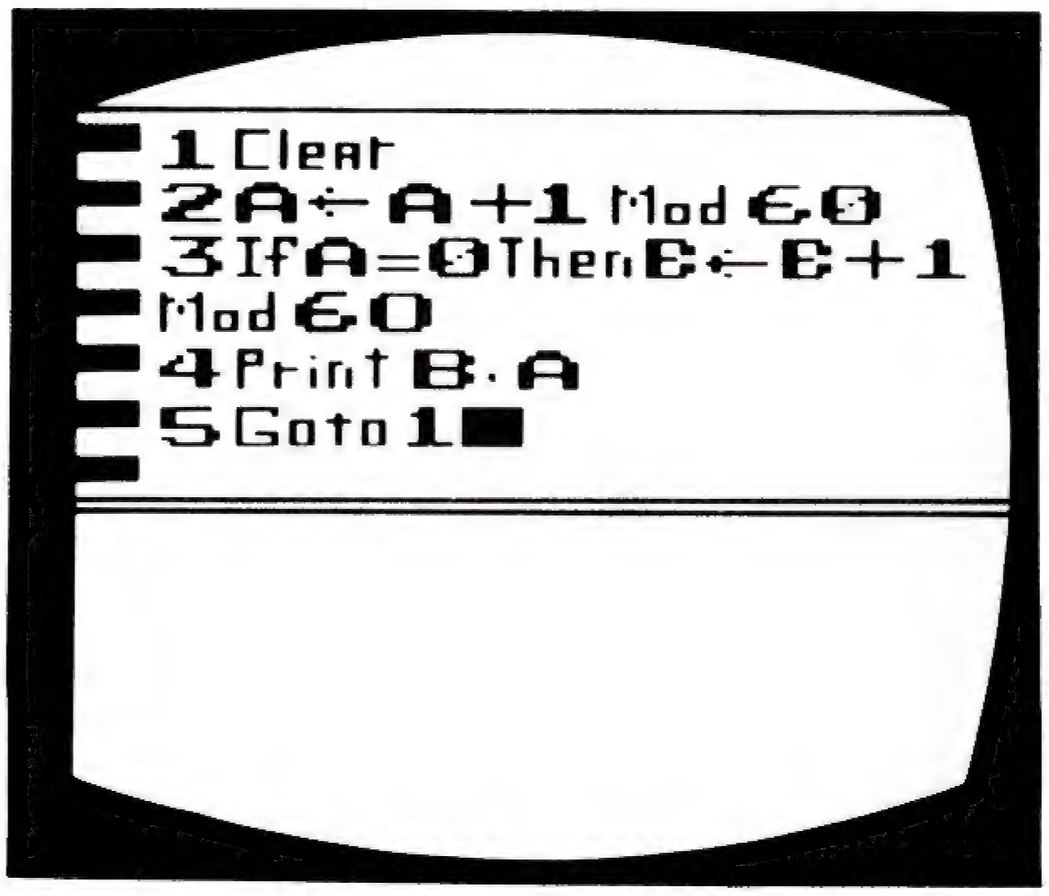
```
1 Note←Note+Key+
7
2 Goto 1■
```

```
1 A←Key
2 If A>0 Then Note←A
—1
3 Goto 1■
```

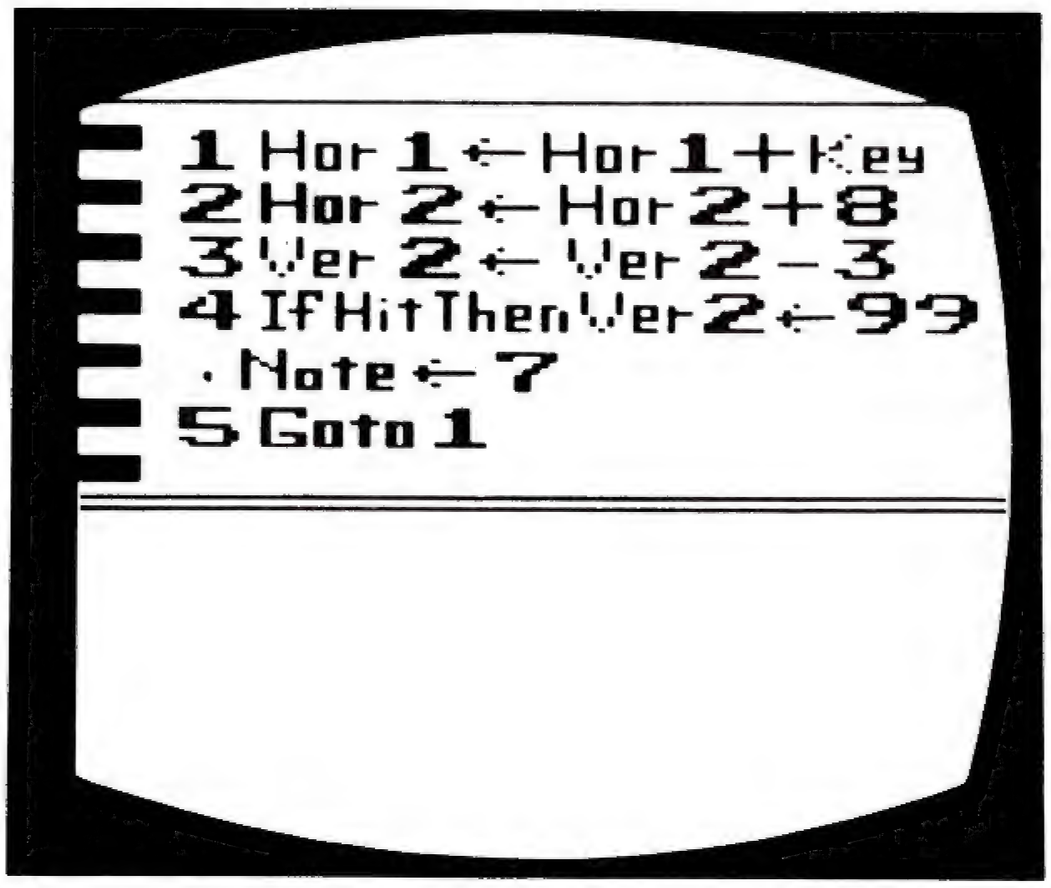
```
1 A←A+1
2 Note←A
3 If (A Mod 2)=0 Then
Note←0 Else Note←4
4 Goto 1■
```

CLOCK-LIKE PROGRAM

When running the Clock program bring up the **OUTPUT** region only. The **SPEED** may be set at 30 or 60.



PONG® GAME (BALL & PADDLE)



PONG® GAME (without sound)

2 · VERSION FRANÇAISE

1 JEU VIDEO

BASIC PROGRAMMING

INTRODUCTION

BASIC PROGRAMMING (Programmation en BASIC) est un outil destiné à l'enseignement des principales étapes de la programmation des ordinateurs. BASIC est l'acronyme de "**B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode." C'est un langage conçu pour permettre d'apprendre sans peine à "rédiger" des programmes d'ordinateur.

Les programmes d'ordinateur sont de simples listes d'instructions. Ils règlent la circulation des données dans l'ordinateur. La cartouche BASIC PROGRAMMING permet de donner au Video Computer System™ les instructions dont il a besoin pour effectuer certaines tâches simples.

Il ne faut pas oublier que le BASIC PROGRAMMING a une capacité de mémoire limitée par comparaison avec des systèmes informatiques plus complexes. Il constitue néanmoins un excellent outil didactique pour apprendre les notions essentielles de la programmation des ordinateurs.

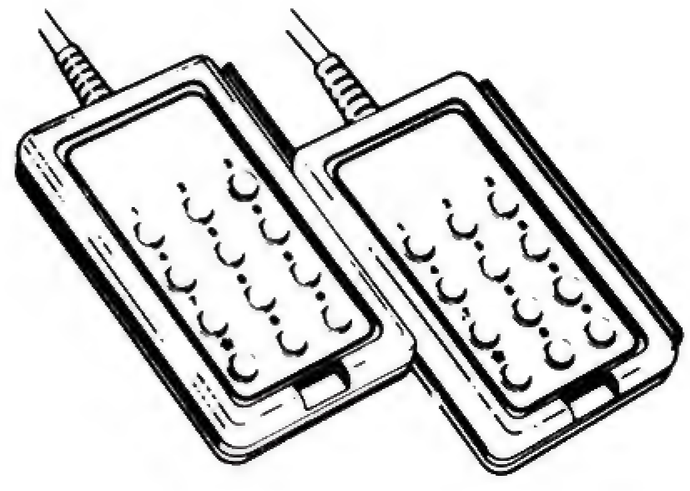
N.B. Avant d'introduire ou de retirer une cartouche ATARI **Game** Program, il faut toujours fermer l'interrupteur du pupitre (position **off**) afin de protéger les composants électroniques et d'éviter une usure prématurée du Video Computer System ATARI.

Cette cartouche risque de provoquer un "défilement" de l'image sur certains téléviserus. Dans ce cas, il pourra être nécessaire de régler la STABILITÉ VERTICALE de l'image.

COMMANDES À CLAVIER

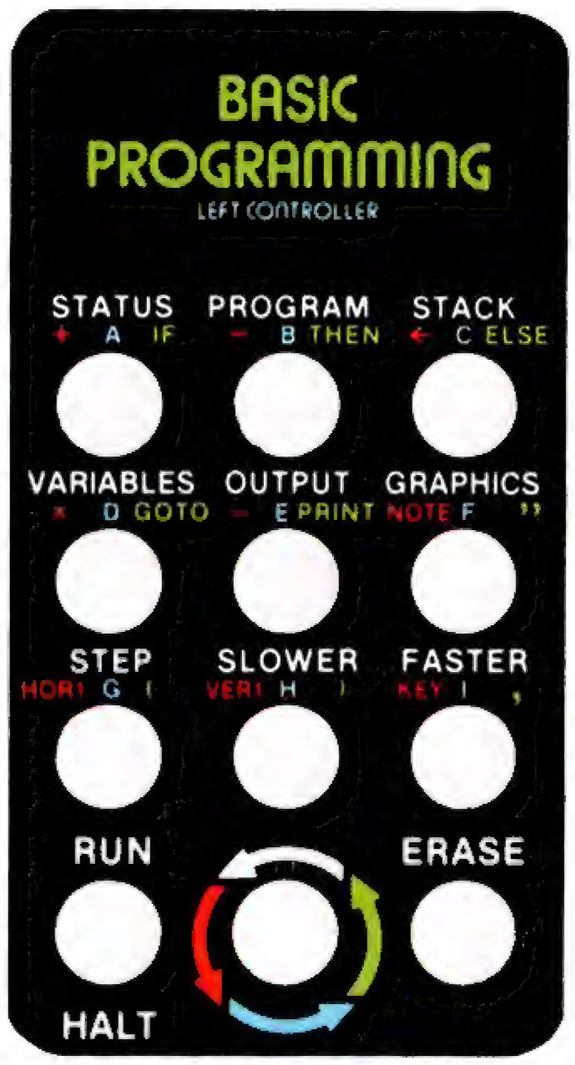
Avec cette cartouche ATARI® Game Program™, utiliser les commandes à clavier. S'assurer que leurs câbles sont bien branchés sur les prises de COMMANDE situées à l'arrière du Video Computer System™ ATARI.

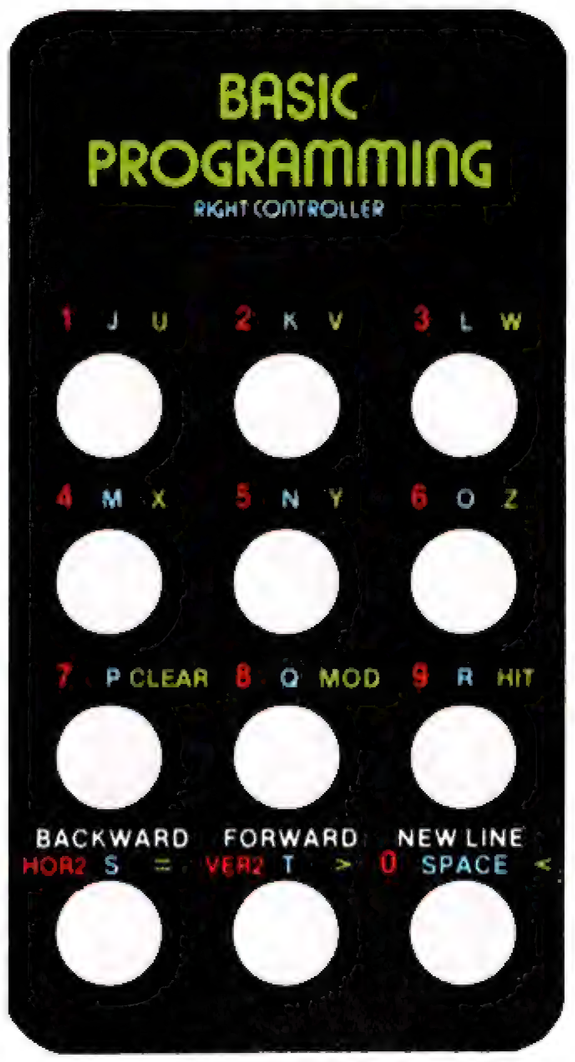
Pour plus de détails, se reporter à la section 3 de la notice du Video Computer System.



Pour relier les deux commandes, glisser la languette de la commande de gauche dans la rainure de la commande de droite. Ainsi réunies, les deux commandes forment un clavier de 24 touches qui permet de composer un programme et de commander l'affichage sur l'écran du téléviseur.

Sortir les étiquettes des commandes à clavier de leur enveloppe. Placer l'étiquette marquée **LEFT** (gauche) sur le clavier branché sur la prise de la **COMMANDE DE GAUCHE** située à l'arrière du pupitre; l'étiquette marquée **RIGHT** (droite) sur le clavier branché sur la prise de la **COMMANDE DE DROITE.**





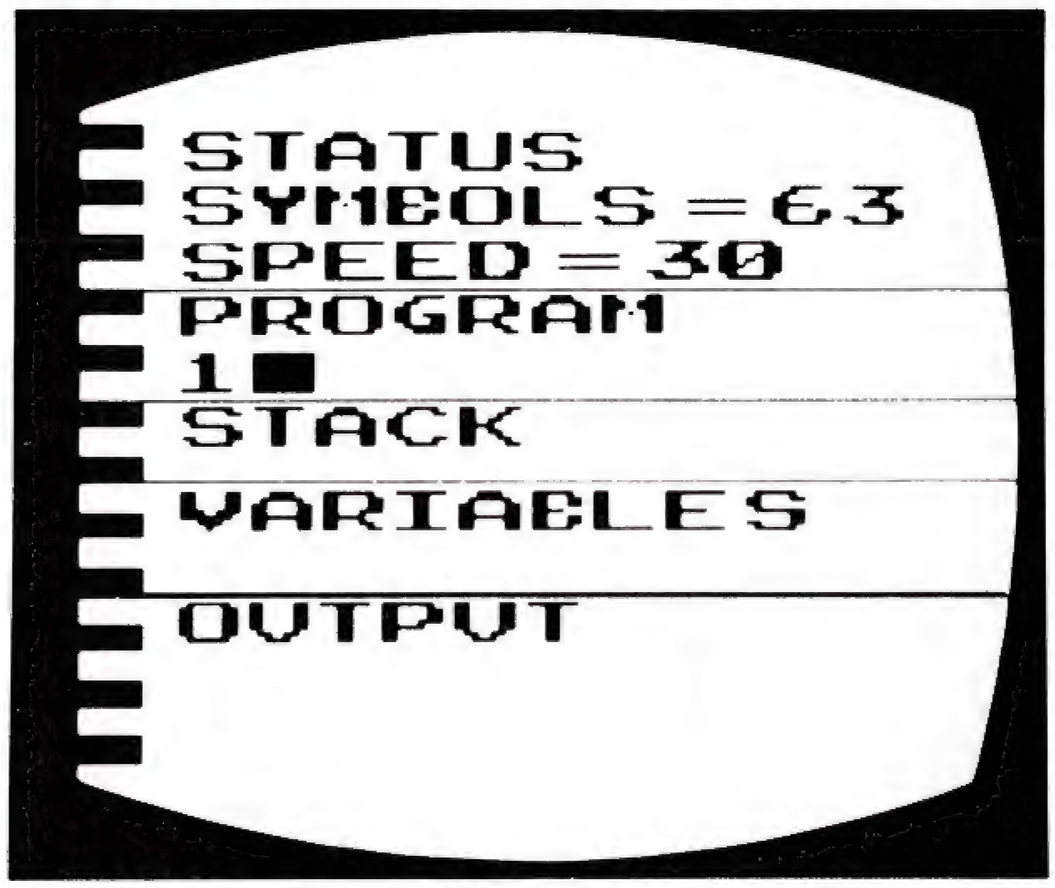
ZONES D'AFFICHAGE

L'affichage est divisé en six zones:

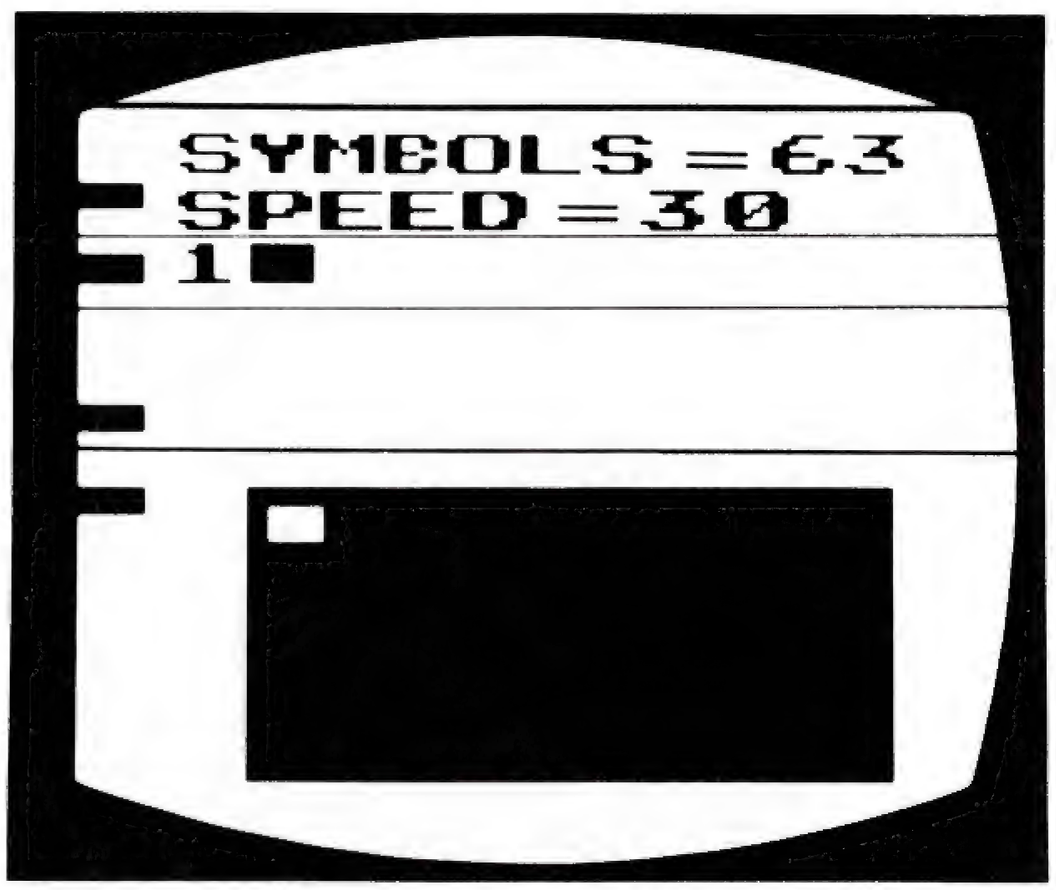
1. La zone **PROGRAM** (programme) qui sert à donner des instructions à l'ordinateur.
2. La zone **STACK** (pile) où s'inscrivent des résultats provisoires pendant que l'ordinateur exécute le programme.
3. La zone **VARIABLES** qui indique la valeur de chacune des variables pendant l'exécution du programme.
4. La zone **OUTPUT** (sortie) qui indique les résultats obtenus par l'exécution du programme.
5. La zone **STATUS** (état) qui indique à tout moment la capacité de mémoire disponible pour le programme. Cette zone indique également la vitesse d'exécution du programme.
6. La zone **GRAPHICS** (graphiques) qui comporte deux carrés de couleur dont le déplacement est commandé par le programme.

Avant de commencer le programme, mettre le **sélecteur de difficulté de gauche (left difficulty)** en position **b.** On verra ainsi où se trouvent les diverses zones de l'affichage. Lorsqu'on met le

sélecteur de difficulté de gauche en position **a**, les titres de chaque zone disparaissent de l'affichage et la zone GRAPHICS apparaît.



Avec cette cartouche, on n'utilise pas le **sélecteur de difficulté de droite (right dificulty).**



LE CURSEUR

Placer le **sélecteur de difficulté de gauche (left difficulty)** en position **b**, puis couper et rétablir l'alimentation du pupitre (position **off** suivie de **on**). Dans la zone **PROGRAM** se trouve un rectangle blanc: c'est le curseur. Repérer la touche de commande de décalage (*Shift*) qui se trouve au centre de la rangée inférieure de la commande de gauche. La frapper quatre fois de suite: la couleur du curseur passe successivement du blanc au rouge, du rouge au bleu, du bleu au vert et du vert au blanc.

Le curseur sert à introduire le programme. Les quatre couleurs de la touche de décalage correspondent à la couleur des commandes ou entrées du clavier. Le mode blanc sert à donner des ordres à l'ordinateur, les autres modes à introduire des symboles dans le programme.

DÉPLACEMENT DU CURSEUR D'UNE ZONE À L'AUTRE

S'assurer que le **sélecteur de difficulté de gauche (left difficulty)** est en position **b**, puis couper et rétablir l'alimentation du pupitre (position **off** puis **on**). Frapper la touche **FORWARD** (avance): le curseur passe de la zone **PROGRAM** à la zone **STACK**. Frapper de nouveau la touche **FORWARD**: le curseur passe de la zone **STACK** à la zone **VARIABLES**. Frapper la touche **FORWARD** une troisième fois: le curseur passe de la zone **VARIABLES** à zone **OUTPUT**.

Pour ramener le curseur dans la zone **PROGRAM**, frapper la touche **BACKWARD** (retour). On peut frapper les touches **FORWARD** et **BACKWARD** une fois pour chaque changement de zone, ou bien on peut les maintenir enfoncées.

Mettre ensuite le **sélecteur de difficulté de gauche** en position **a.** Les titres des zones disparaissent et une partie de la zone **GRAPHICS** apparaît. Faire de nouveau passer le curseur d'une zone à l'autre. On remarquera qu'il n'entre pas dans la zone **GRAPHICS**.

SUPPRESSION DE L'AFFICHAGE DES ZONES

Mettre de nouveau le **sélecteur de difficulté de gauche** en position **b,** puis couper et rétablir l'alimentation du pupitre (position **off** suivie de **on**). Sur le côté gauche du clavier se trouve une série de commandes (mode blanc) qui correspondent à chacune des zones: **STATUS, PROGRAM, STACK, VARIABLES, OUTPUT** et **GRAPHICS.** Nous commencerons pas la zone **STATUS**: frapper la touche **STATUS** une fois; la zone **STATUS** disparaît et la zone **PROGRAM** se trouve amenée en haut de l'écran.

Frapper la touche **PROGRAM**: la zone **PROGRAM** disparaît et la zone **STACK** se trouve amenée en haut de l'écran. Frapper la touche **STACK**: la zone **STACK** disparaît et la zone **VARIABLES** se trouve amenée en haut de l'écran. Frapper la touche **VARIABLES**: la zone **VARIABLES** disparaît, ce qui amène la zone **OUTPUT** en haut de l'écran.

Faire disparaître la zone **OUTPUT** en frappant la touche **OUTPUT**. La zone **GRAPHICS** est alors entièrement visible sur l'écran. Frapper la touche **GRAPHICS** pour faire disparaître cette zone. Aucune zone ne doit alors être visible sur l'écran.

Frapper alors la touche **STACK**. La zone **STACK** reparît sur l'écran. La faire disparaître et amener les zones **OUTPUT** et **VARIABLES**. On peut afficher et faire disparaître toute zone désirée, que le **sélecteur de difficulté de gauche** se trouve en position **a** ou **b**.

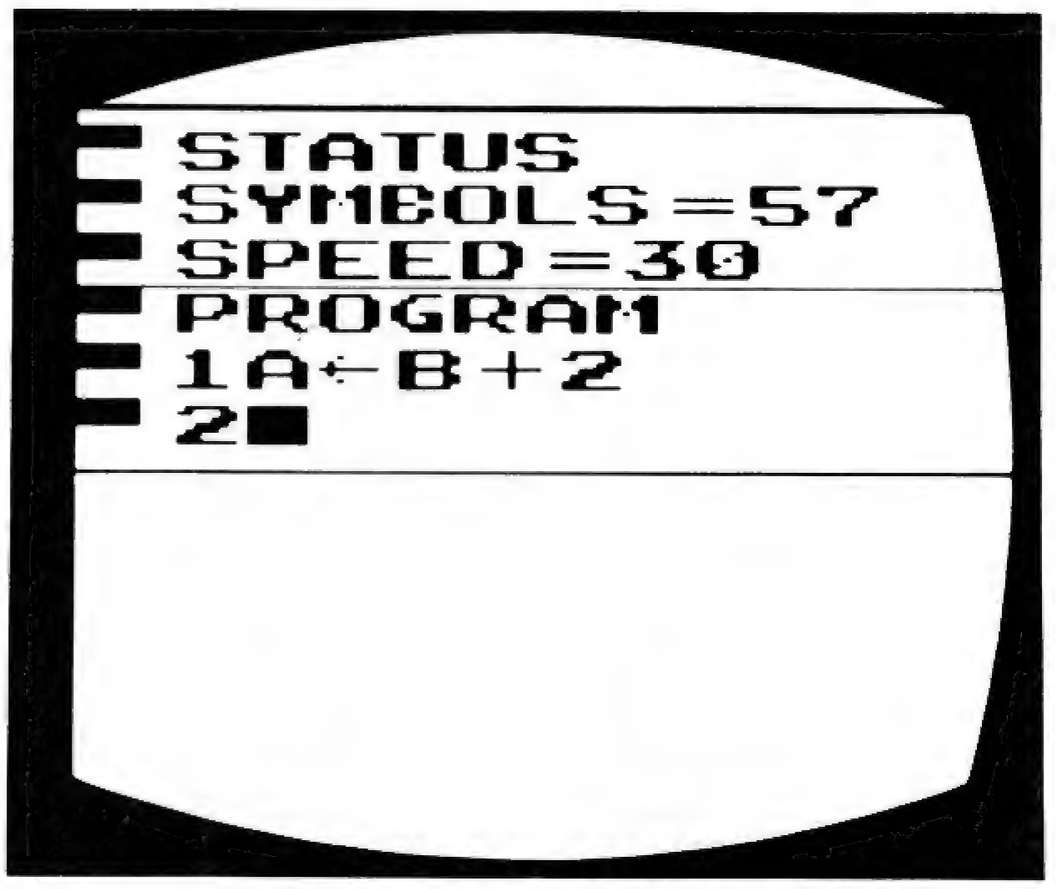
On notera que l'exécution d'un programme n'est pas affectée par l'affichage ou le non-affichage des diverses zones.

EXÉCUTION D'UN PROGRAMME

Exécutons un programme simple. S'assurer que le **sélecteur de difficulté de gauche (left difficulty)** est en position **b**; couper et rétablir l'alimentation du pupitre (position **off** puis **on**). (Une autre manière d'effacer un programme et de mettre toutes les valeurs à zéro consiste à presser le **sélecteur de jeu (game select)**. Si l'on presse le bouton de **remise à zéro (game reset)**, on ne fait qu'effacer les valeurs et ramener le programme à son début, sans l'effacer.)

Faire disparaître de l'écran les zones **STACK, VARIABLES, OUT-PUT** et **GRAPHICS**. Le curseur se trouve alors en mode blanc et à la droite du numéro **1** dans la zone **PROGRAM**. Amener le curseur en mode bleu et frapper la touche **A**. La lettre **A** apparaît sur l'écran à côté du **1**. (Chaque ligne est numérotée, ce qui permet de voir où elle finit et où commence la suivante.) Amener ensuite le curseur en mode rouge et frapper la touche ←. Une petite flèche apparaît à côté du **A**. Revenir au mode bleu et entrer un **B**. Faire passer le curseur au rouge et entrer **+** et le nombre **2**. Revenir ensuite au mode blanc et entrer **NEW LINE** (à la ligne). Il faut toujours être en mode blanc pour aller à la ligne.

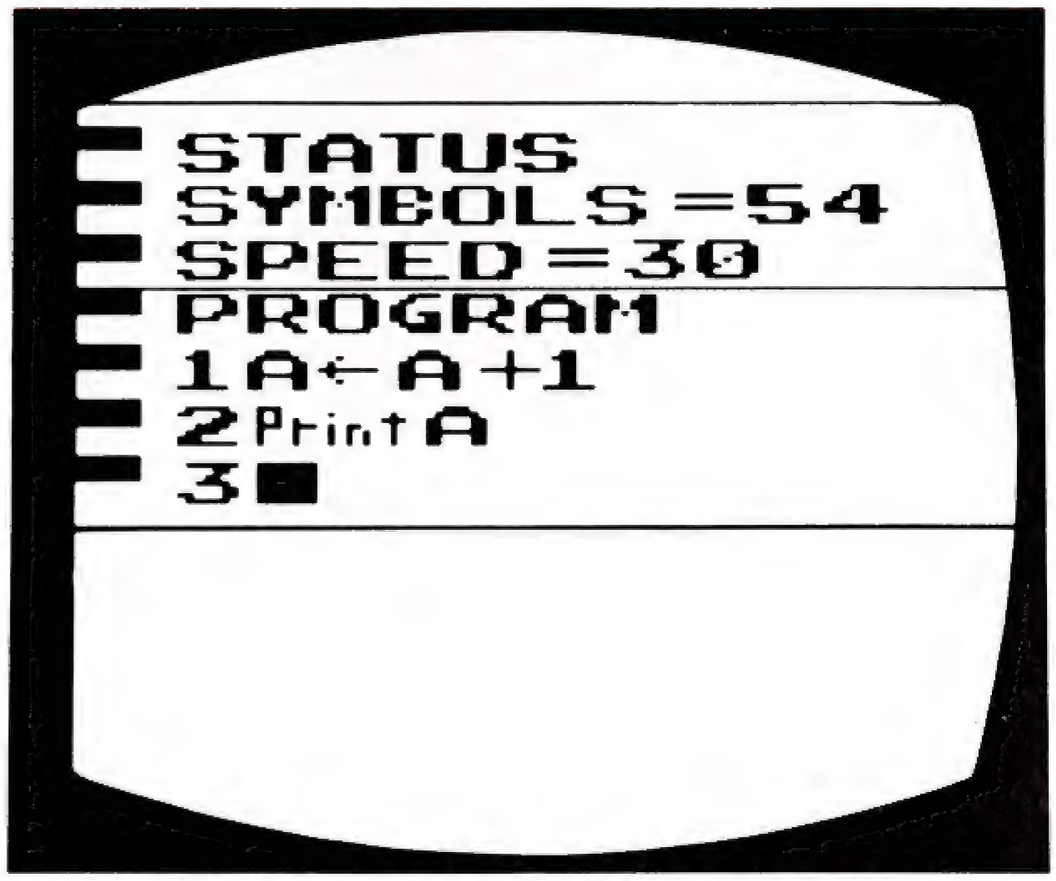
L'affichage doit se présenter ainsi:



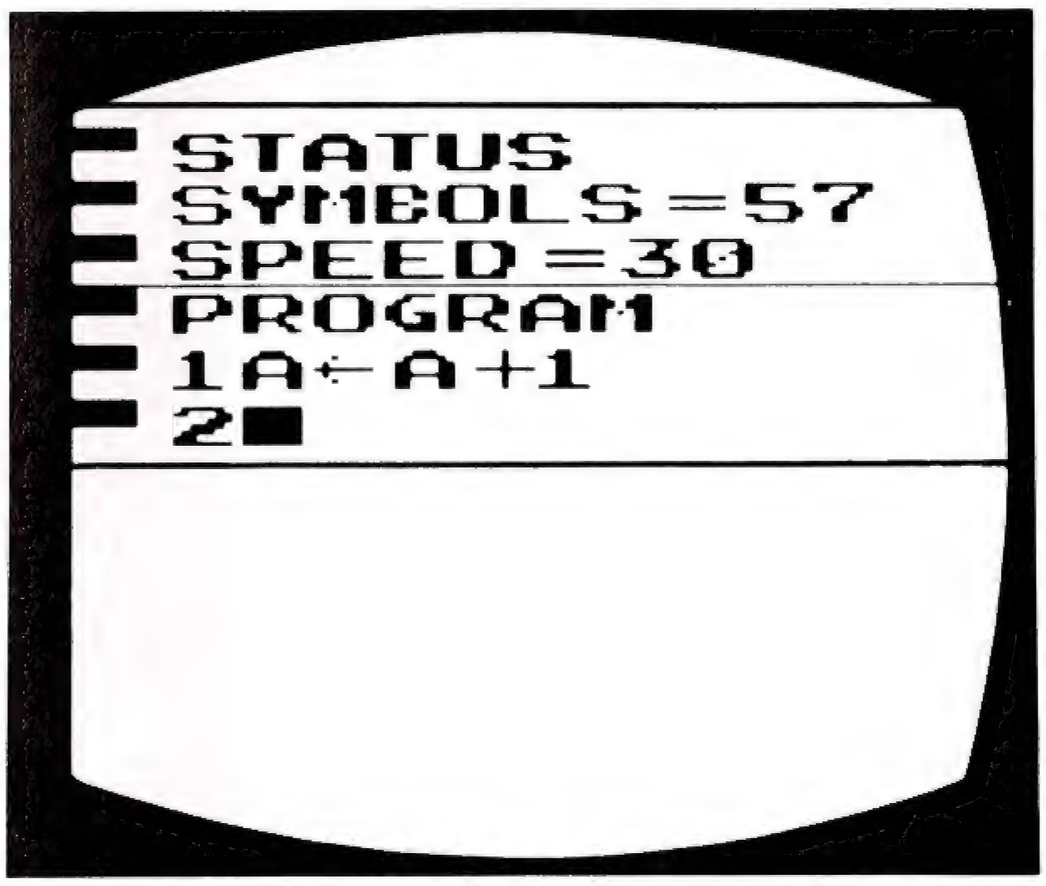
Si l'on fait une erreur dans l'introduction du programme, on peut l'effacer à l'aide de la touche **ERASE** (effacement). On notera que cette touche **n'a pas** de couleur-code. On peut l'utiliser quelle que soit la couleur du mode de la touche de décalage.

Faisons un essai avec la ligne que l'on vient d'introduire dans le programme. Frapper la touche **ERASE** une seule fois. Le curseur passe immédiatement de la ligne **2** à la droite du chiffre **2** qui se trouve sur la ligne **1**. Frapper de nouveau la touche **ERASE**: le **2**

est effacé du programme. Entrer alors **1** en utilisant le mode rouge. Amener le curseur en mode blanc et, à l'aide de la touche **BACKWARD**, l'amener immédiatement à la droite du **B** inscrit dans le programme. Frapper la touche **ERASE**: la lettre disparaît du programme. La remplacer par un **A** (mode bleu). Utiliser la touche **FORWARD** en mode blanc pour aller en fin de ligne **1** et entrer **New Line**.

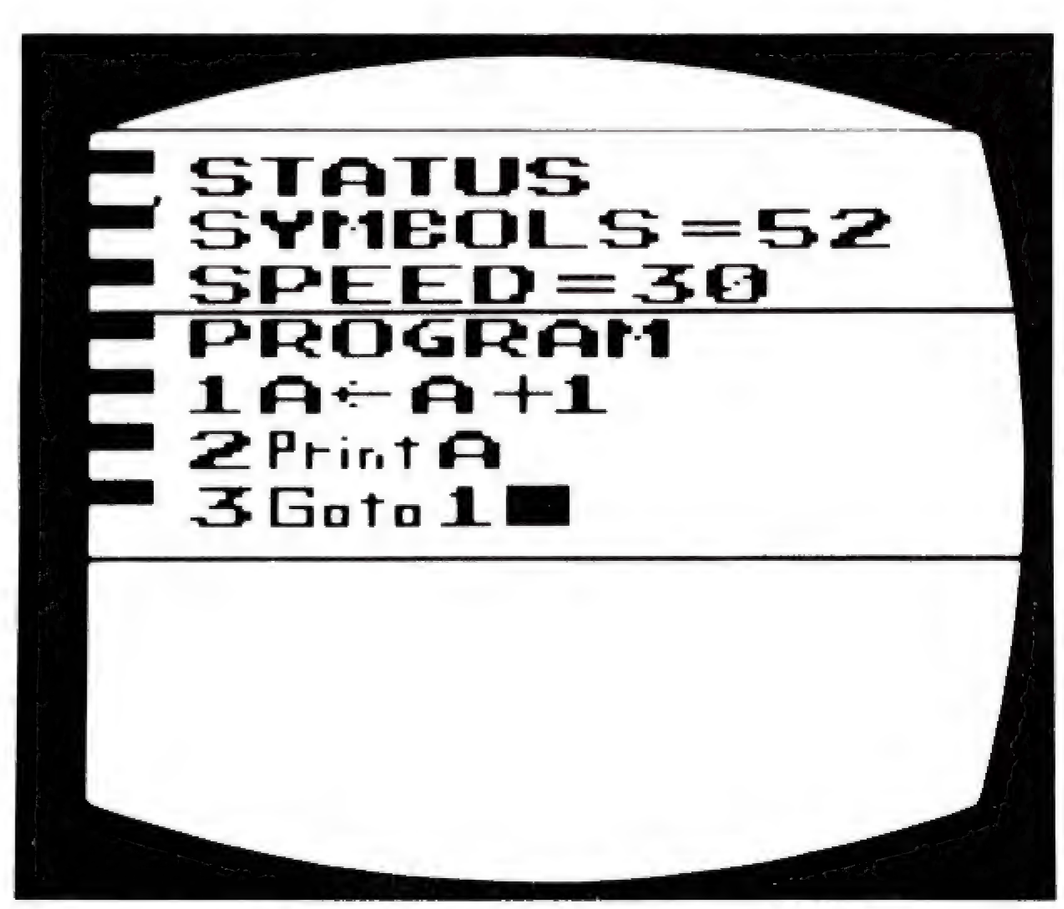


Ne pas oublier que le curseur ne doit pas se trouver sur le symbole à effacer, mais immédiatement à sa droite. L'affichage se présente alors ainsi:



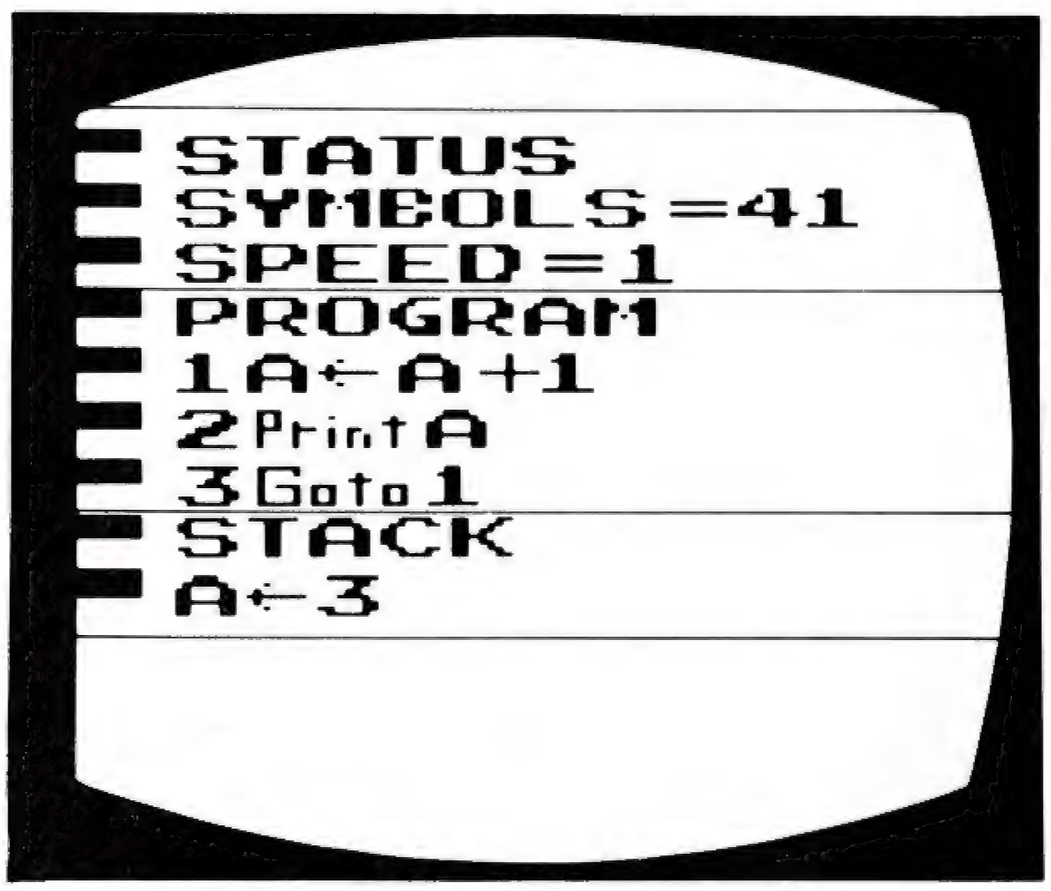
Le curseur se trouve immédiatement à la droite du **2** de la ligne **2**. Se mettre en mode vert et entrer **PRINT** (impression). Ensuite, en mode bleu, entrer **A** et aller à la ligne (''New Line''). L'affichage se présente alors ainsi:

Ensuite, le curseur étant en mode vert, entrer **GOTO** (aller à) puis, en mode rouge, entrer le chiffre **1**. Avant de continuer, regardons l'affichage:



On notera que la zone **STATUS** indique qu'il reste 52 ''octets'' ou symboles de mémoire. La vitesse est fixée à 30 (**SPEED = 30**). Mettre le curseur en mode blanc et frapper la touche **SLOWER** (ralentissement). On notera que la vitesse diminue chaque fois que l'on frappe cette touche. Frapper la touche **FASTER** (accélération) pour augmenter la vitesse.

Avant de commencer le programme, entrer **SPEED = 1**. Frapper ensuite la touche **STACK**. **Frapper** deux fois de suite la touche **RUN/HALT** (exécution/arrêt) et l'exécution du programme commence. On peut voir comment l'ordinateur en exécute les différentes parties.



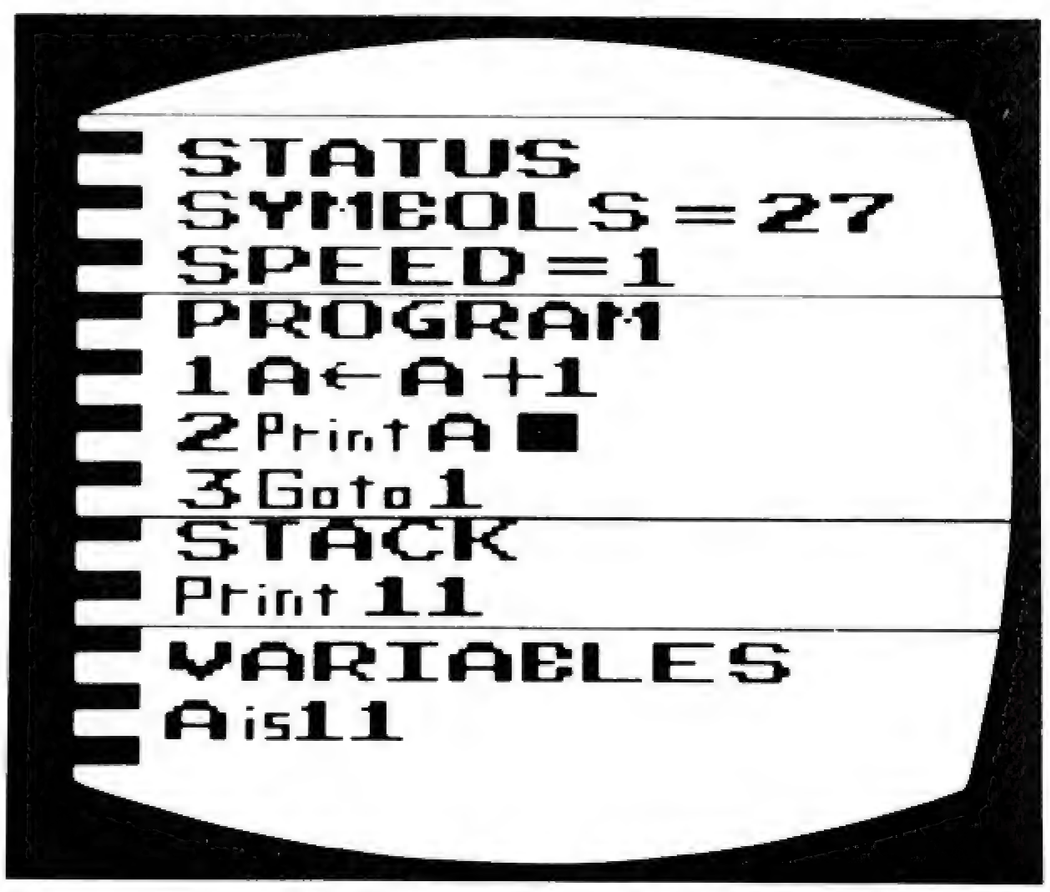
Pour arrêter le programme, frapper **RUN/HALT**. Comme indiqué précédemment, le bouton de **remise à zéro (game reset)** permet d'effacer toutes les valeurs du programme (sans effacer le programme proprement dit) et de ramener le programme à son début.

Analysons le programme pas à pas au fur et à mesure que l'ordinateur l'exécute. Mettre la vitesse **(SPEED)** à **60**. Le curseur étant en mode blanc, frapper la touche **STEP** (pas). **Chaque** partie du programme est alors prise pas à pas, chaque fois que l'on frappe cette touche.

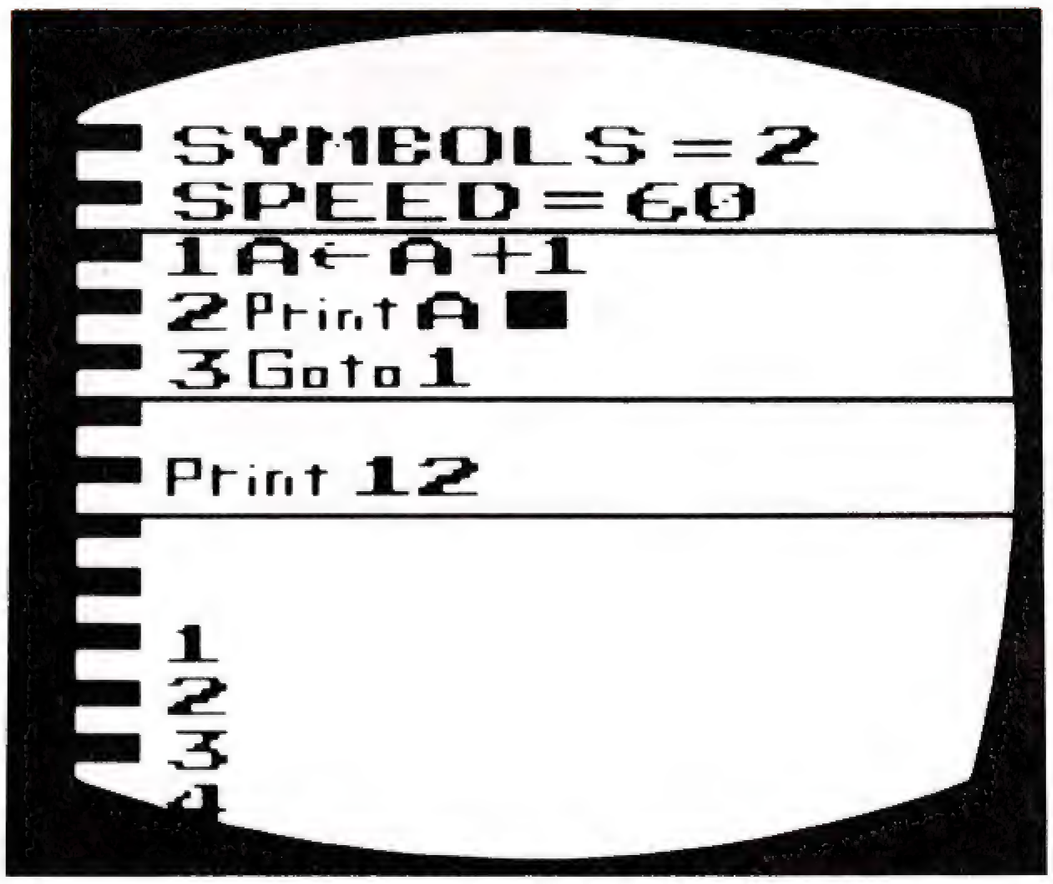
À la ligne **1**, le programme indique: **A ← A + 1**. L'ordinateur lit: **A** ''devient'' **A + 1**. Observer la zone **STACK** lors de l'exécution de la ligne 1. La ligne 2 ordonne à l'ordinateur d'imprimer A **(Print A)**, c'est-à-dire d'imprimer la valeur de A (calculée à la ligne 1) dans la zone **OUTPUT**.

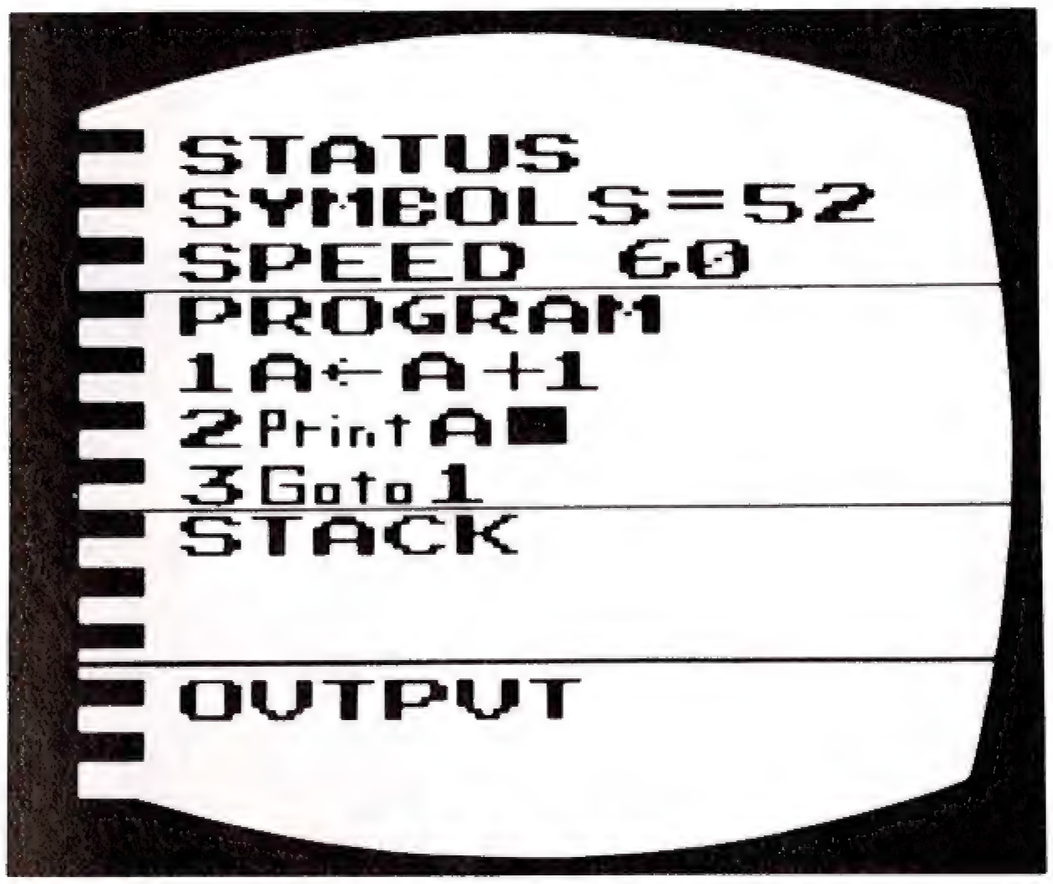
Le principe de l'impression sera expliqué plus loin. La ligne **3** ordonne à l'ordinateur d'aller à 1 **(Goto 1)**. L'ordinateur doit revenir à la ligne **1** et trouver une nouvelle valeur de **A**. Chaque fois qu'on exécute le programme, l'ordinateur prend une nouvelle valeur de **A**, l'imprime et retourne à la ligne **1**. Il continue à exécuter le programme de cette manière jusqu'à ce que la ''mémoire'' soit épuisée. La capacité de mémoire restante est affichée dans la section **SYMBOLS** de la zone **STATUS**.

Afficher la zone **VARIABLES**. Réduire la vitesse à **1 (SPEED + 1)** et effacer les valeurs du programme en pressant le bouton de **remise à zéro (game reset)**. Frapper la touche **RUN/HALT** et observer comment l'ordinateur exécute le programme. À chaque pas du programme, l'ordinateur indique la valeur de **A**. L'affichage se présente ainsi:



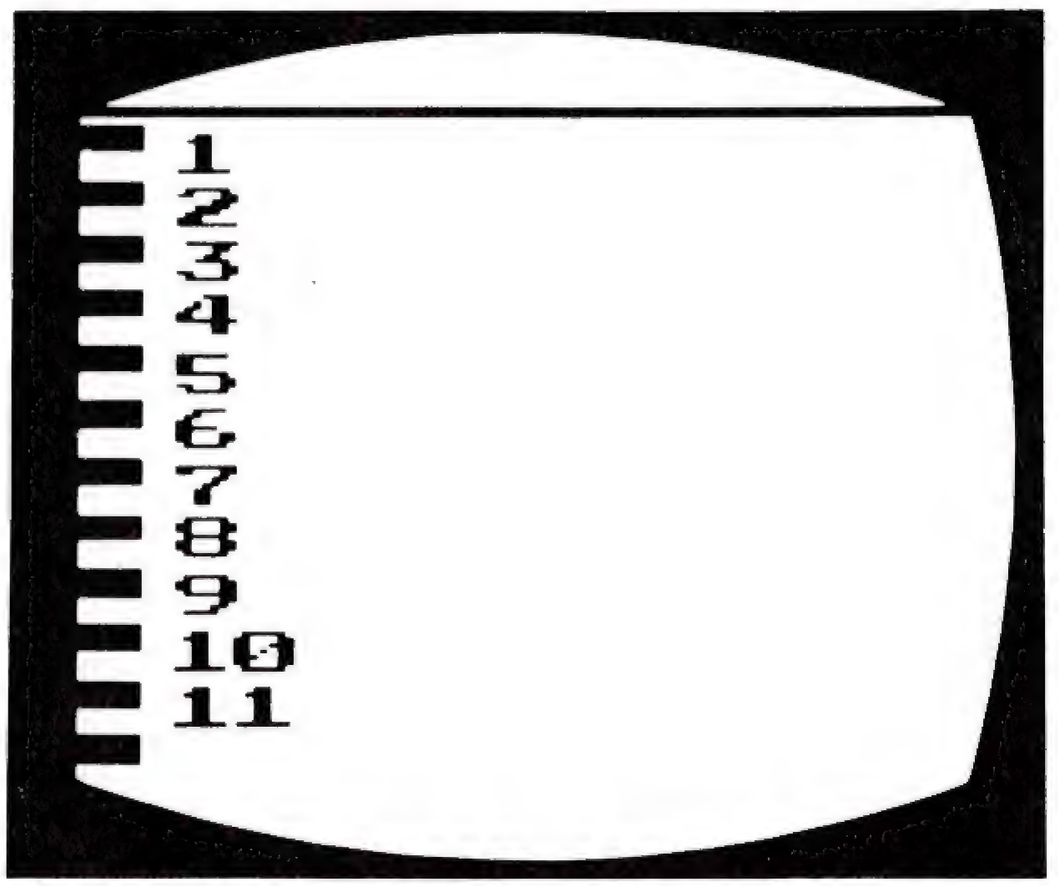
Interrompre le programme et effacer les valeurs **(remise à zéro)**. Faire disparaître la zone **VARIABLES** et amener la zone **OUTPUT** sur l'écran. Régler la vitesse à 60 **(SPEED = 60)**. L'affichage se présente alors ainsi:



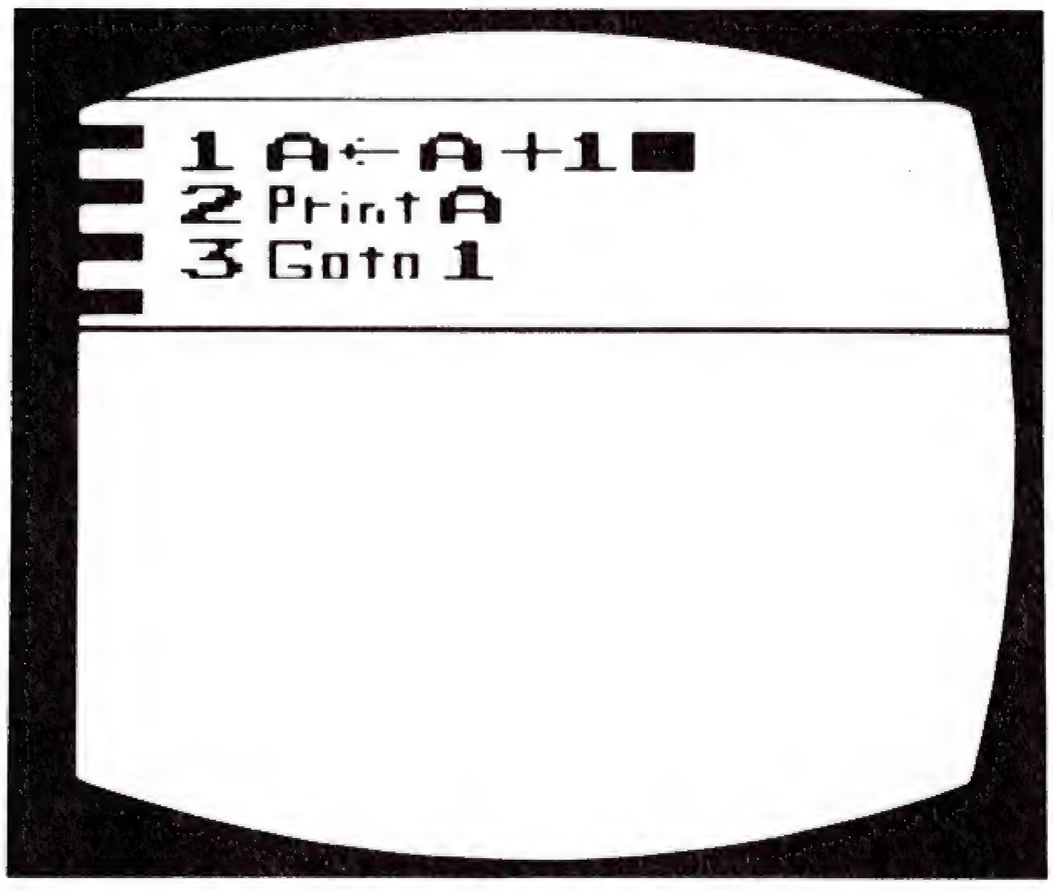


Est-il possible de consacrer davantage de place à l'affichage de la zone **OUTPUT**? Faire disparaître les zones **STATUS, PROGRAM** et **STACK** de l'écran. L'affichage se présente ainsi:

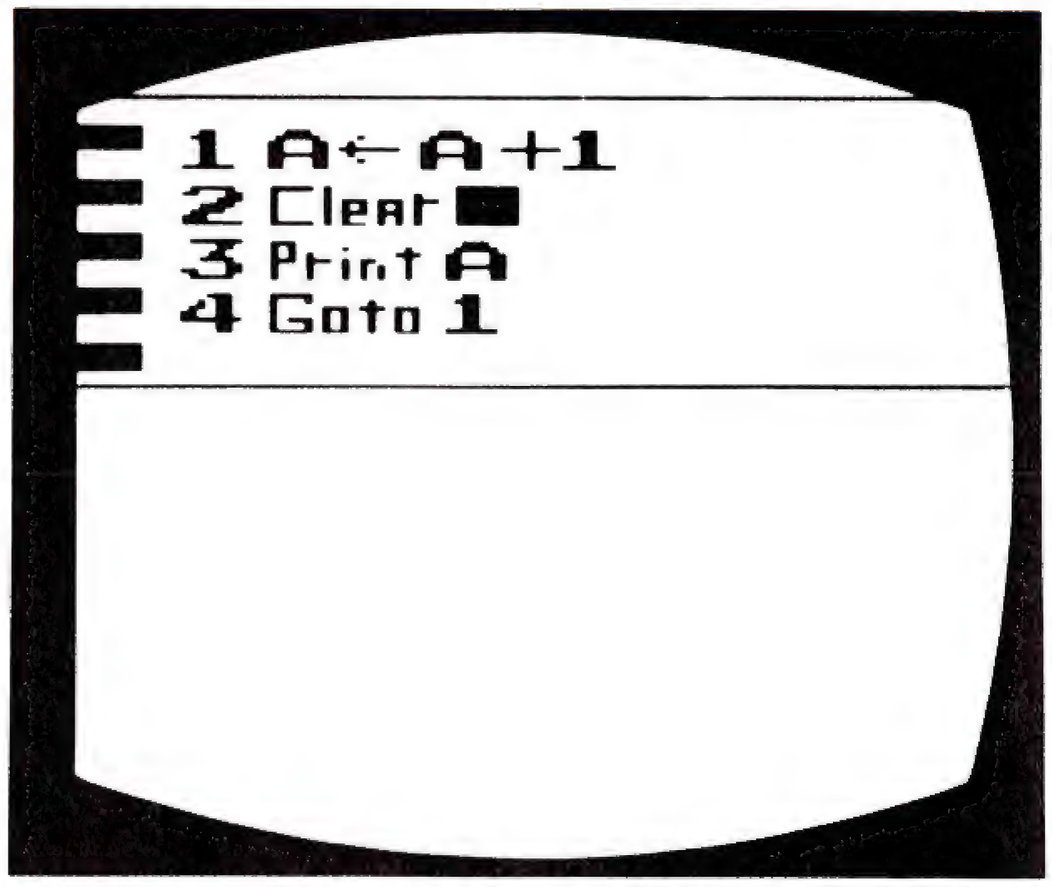
Frapper la touche **RUN/HALT** et commencer l'exécution du programme. À la ligne **2**, l'ordinateur reçoit l'ordre d'imprimer **A (PRINT A)**. Il imprime alors la valeur actuelle de **A** dans la zone **OUTPUT**. Observer la section **SYMBOLS** de la zone **STATUS**. Bien que la zone **OUTPUT** affiche 1, l'ordinateur n'a pas fini d'imprimer la valeur variable de **A**. Mettre le **sélecteur de difficulté de gauche (left difficulty)** en position **a**. L'affichage se présente alors à peu près ainsi:



Comment faire pour ne pas surcharger la zone **OUTPUT** du programme? Interrompre le programme et effacer les valeurs à l'aide du **bouton de remise à zéro**. Faire disparaître la zone **OUTPUT** de l'écran et amener la zone **PROGRAM**. À l'aide de la touche **FORWARD**, amener le curseur à la fin de la ligne **1**; l'affichage se présente alors ainsi:



Frapper alors la touche **NEW LINE**. On introduit ici un nouvel ordre. Entrer **CLEAR** (effacement) (mode vert). L'affichage se présente alors ainsi:



Amener les zones **STACK, VARIABLES** et **OUTPUT**. Laisser le **sélecteur de difficulté de gauche** en position **a** et démarrer le programme. Lorsqu'il arrive à la ligne **2 CLEAR**, il efface la valeur de **A** dans la zone **OUTPUT** et imprime ensuite la valeur suivante de **A** à la ligne **3**.

La cartouche BASIC PROGRAMMING ne fonctionne qu'avec des nombres de deux chiffres; dès qu'on atteint **99** la "boucle est bouclée" et les valeurs de **A** repartent à zéro.

UTILISATION DE LA FONCTION NOTE

Chaque fois que le programme enregistre un chiffre dans NOTE (mode rouge), le haut-parleur du téléviseur émet une note musicale.

Faisons quelques démonstrations à l'aide de programmes. Si le Video Computer System contient déjà des programmes, presser le **sélecteur de jeu (game select)** du pupitre.

Entrer les lignes suivantes:

1 Note ← Note + 1
2 Goto 1 (aller à 1)

Démarrer ensuite le programme. Le ralentir et observer comment il s'exécute dans la zone **STACK**. Observer aussi qu'il imprime la

valeur actuelle de **NOTE** dans la zone **VARIABLES**.

Interrompre le programme et introduire une nouvelle ligne **2**. (En mode blanc, amener le curseur en fin de ligne 1 et frapper **NEW LINE**. L'ancienne ligne **2** devient la ligne **3**.)

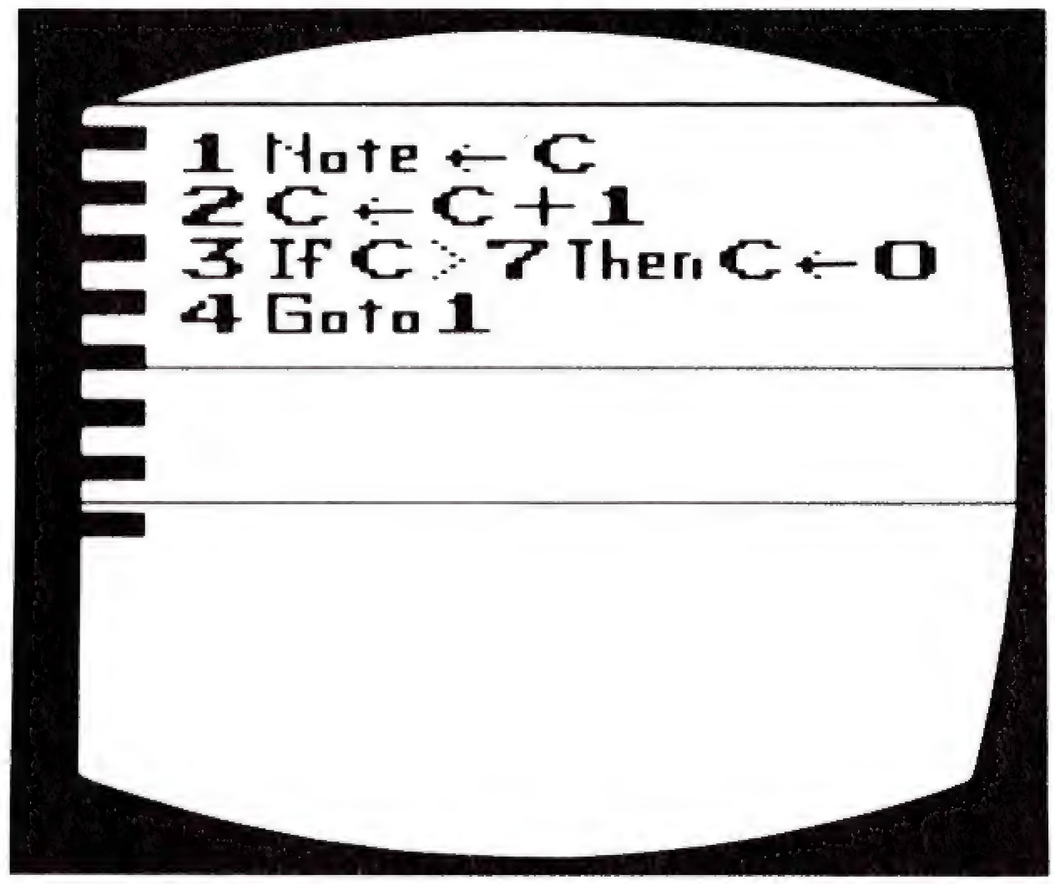
2 If Note > 6 Then Note ← 0
(Si Note > 6 Alors Note ← 0)

L'affichage se présente ainsi:
(aprés enlèvement des zones **STACK** et **VARIABLES**)

Les ordres **IF** et **THEN** (si, alors) indiquent à l'ordinateur que SI une certaine chose se produit, ALORS il doit faire une certaine autre chose. Dans l'exemple, **SI** la variable Note est supérieure à 6, **ALORS** elle doit **prendre** la valeur 0.

Démarrer le programme et observer son exécution dans la zone **STACK**. Observer que lorsque la valeur de Note atteint 7, c'est-à-dire lorsqu'elle dépasse 6 (**< >** 6), le programme lui affecte la valeur 0.

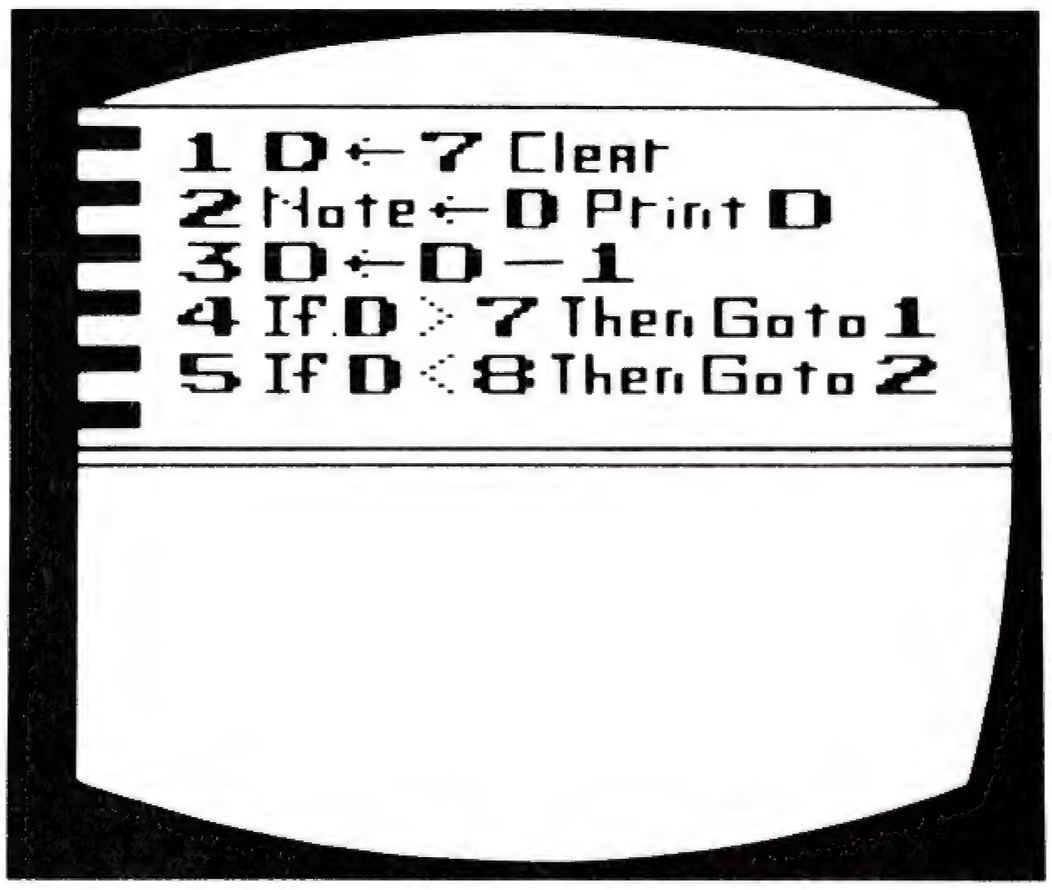
Prenons un autre programme qui donne le même résultat mais d'une manière différente. Entrer ce qui suit:



Exécuter le programme. Observer que le résultat est le même, sauf que les notes sont espacées plus régulièrement. Pour obtenir que le programme affiche la valeur de **C** dans la zone **OUTPUT** entrer les instructions **Print C** (imprimer C) et **Clear** (effacement) en point quelconque du programme. Pour réduire le scintillement de la zone **OUTPUT** qui se produit pendant que le programme indique la valeur demandée, ajouter une virgule (mode vert) aprés la valeur dans la ligne Print, **Print C**.

Dans les programmes, on peut utiliser n'importe quelle lettre **de** l'alphabet pour désigner la variable. Le programme suivant fait appel à toutes les fonctions de touches étudiées jusqu'ici et utilise presque toute la "mémoire" dont dispose l'ordinateur. Après l'avoir introduit, le faire disparaître de l'écran, amener les zones

STATUS, STACK et **OUTPUT**, et exécuter le programme.



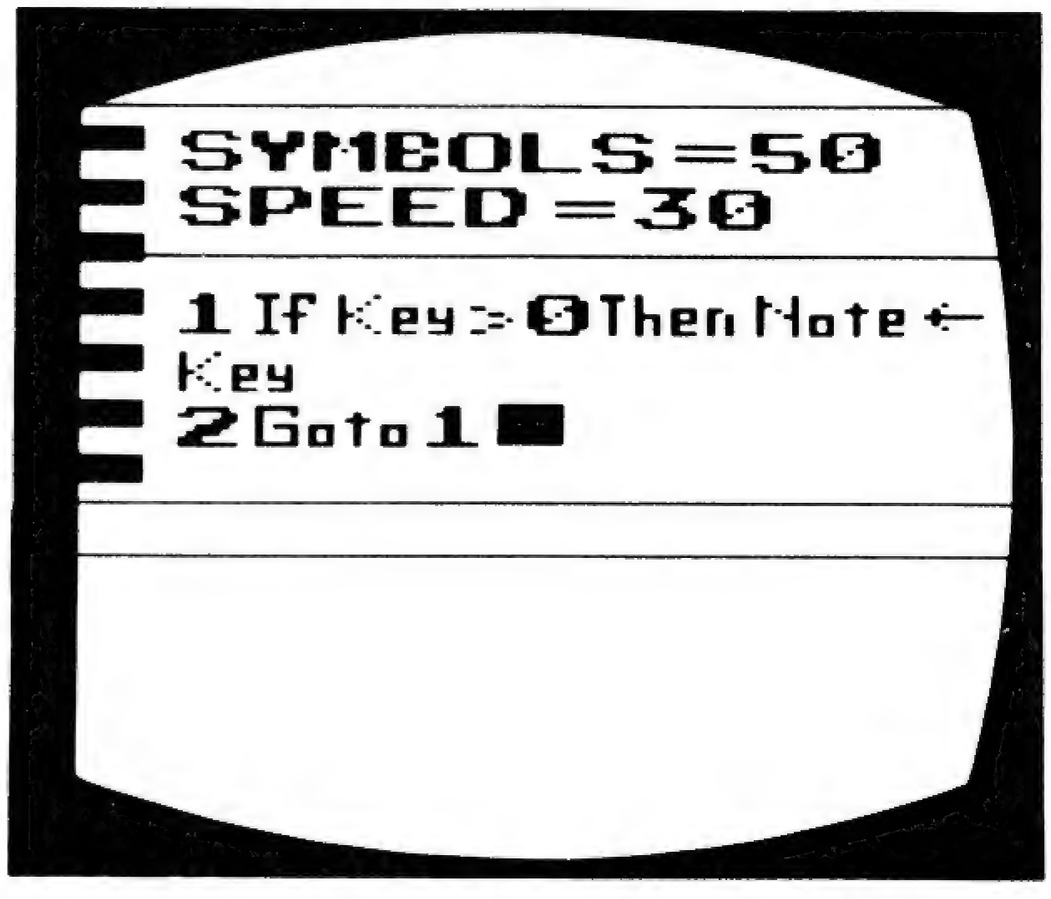
Ce programme comporte deux ordres **IF/THEN** (si/alors). Si les conditions indiquées dans le premier ordre ne sont pas remplies (ligne 4), l'ordinateur passe à la ligne suivante (ligne 5). Si la capacité de mémoire est suffisante, on peut placer plusieurs ordres entre deux ordres **IF/THEN**.

Dans certains cas, on peut placer deux ordres sur la même ligne; voir par exemple les lignes **1** et **2** du programme ci-dessus. De cette manière, on réserve une certaine capacité de mémoire pour la suite du programme.

UTILISATION DES FONCTIONS KEY ET PRINT

La fonction **KEY** (touche) (mode rouge) sert à introduire une variable en cours d'exécution du programme. Le programme évalue **KEY** et la remplace par un nombre que l'on introduit par le côté droit du clavier. Si l'on n'entre aucun nombre, le programme attribue à **KEY** la valeur **0**.

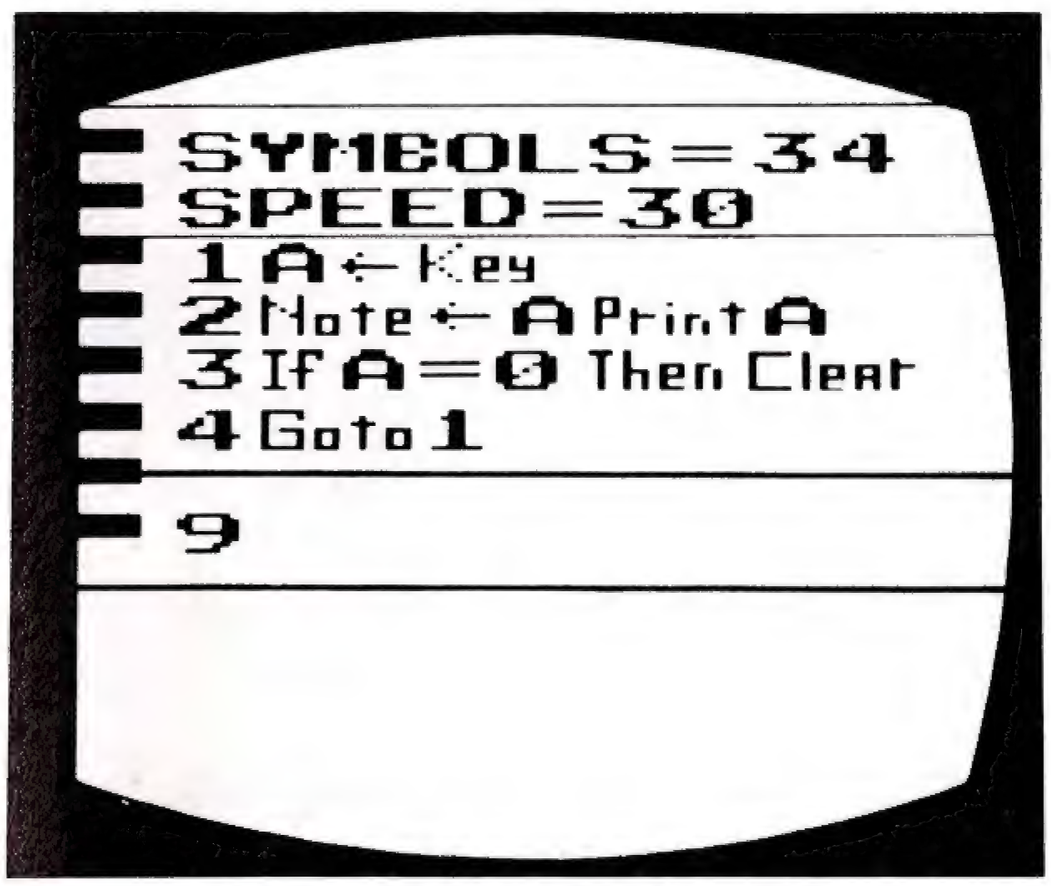
Voici un programme simple qui fait appel à la fonction **KEY**:



Démarrer le programme et frapper certaines des touches du côté droit du clavier. Avec un peu d'entraînement, on arriver à jouer un air.

La fonction **KEY** sert également à programmer dans la zone **GRAPHICS**, comme nous le verrons plus loin.

Essayer le programme suivant qui fait appel aux fonctions **KEY**, **NOTE**, et **PRINT**:

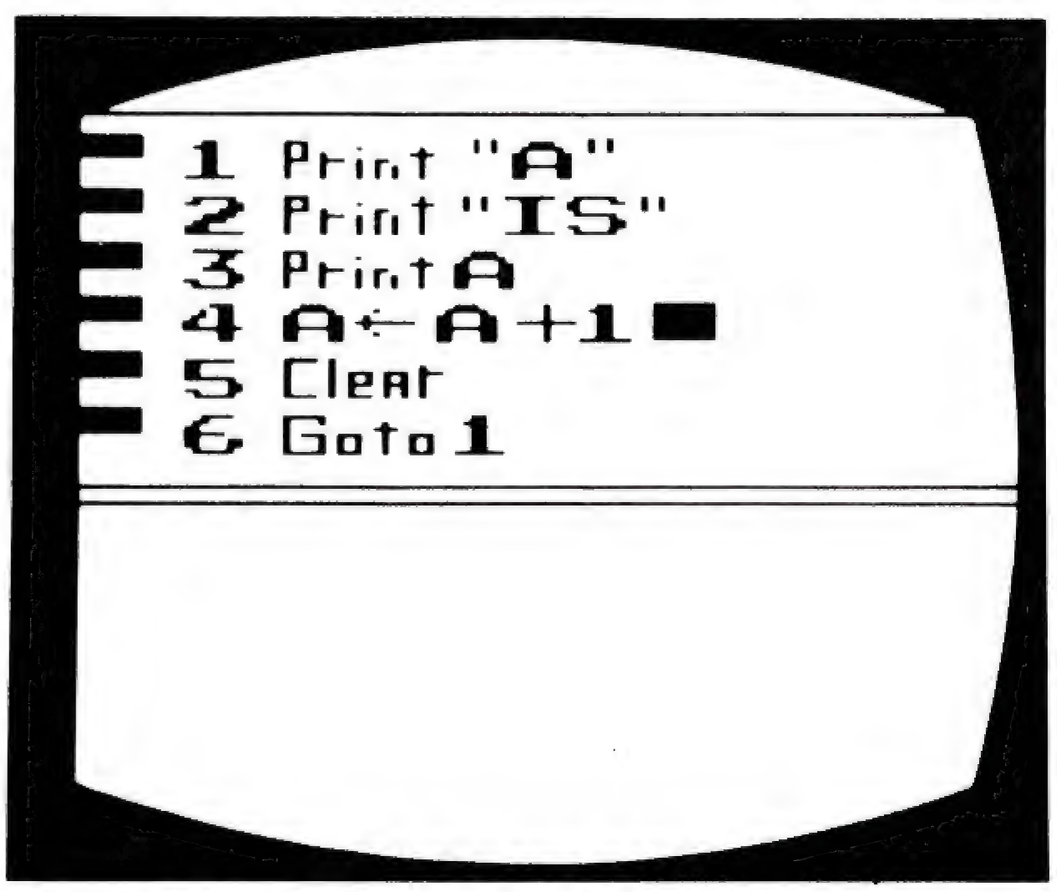


Observer ce programme dans la zone **OUTPUT**. Observer que si l'on entre un nombre par le côté droit du clavier, le programme joue la note correspondante et affiche le nombre dans la zone **OUTPUT**.

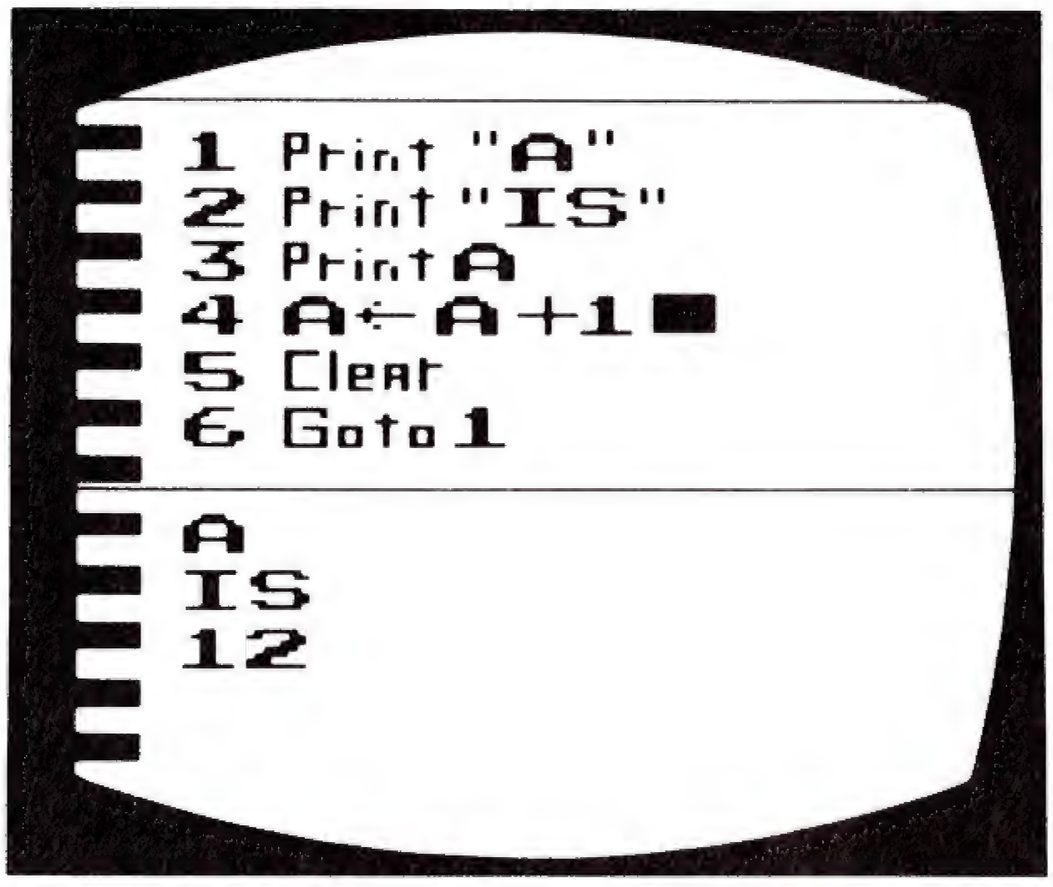
Nous allons maintenant légèrement modifier ce programme. À la ligne **2**, ajouter une virgule (,) après **Print A**, et démarrer le programme. La virgule indique au programme que l'on désire voir autant de variables que possible imprimées sur une même ligne.

UTILISATION DE LA FONCTION PRINT

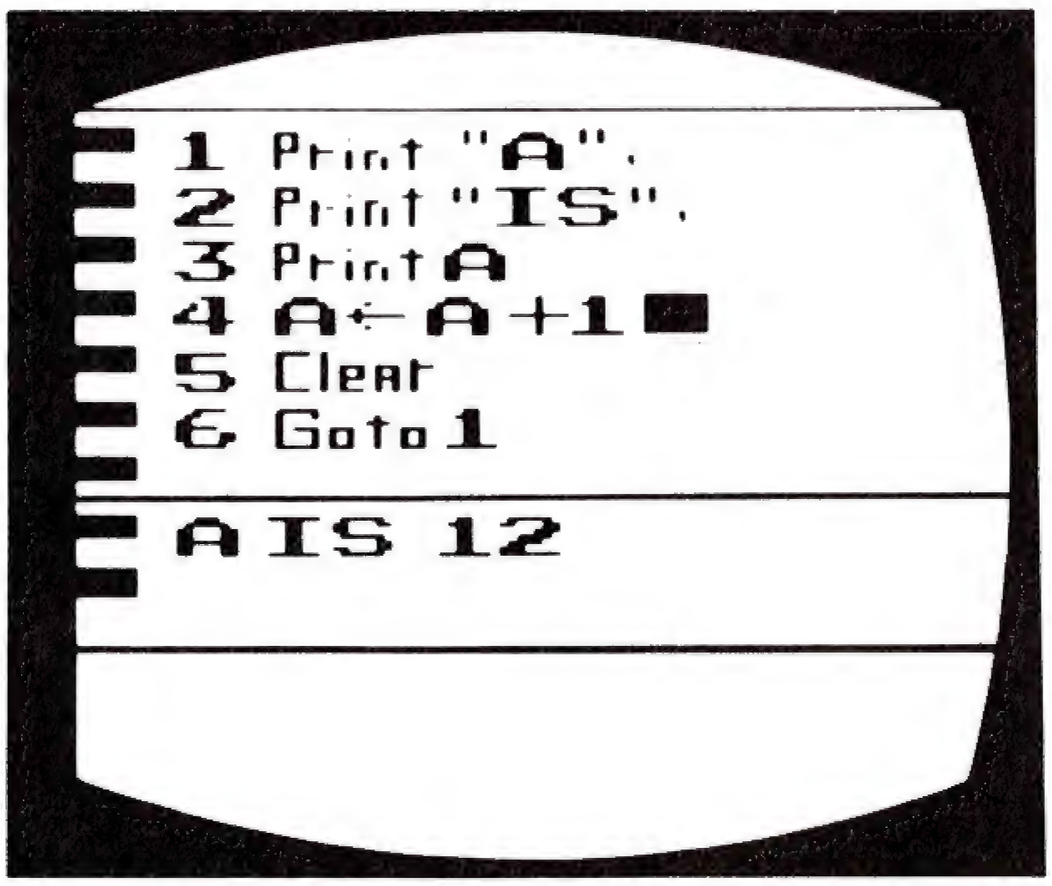
Comme nous l'avons montré dans les programmes qui précèdent, la fonction **PRINT** (impression) sert à ordonner au programme d'imprimer la valeur d'une variable donnée dans la zone **OUTPUT**. La fonction **PRINT** est également utilisée pour ordonner au programme d'imprimer certains mots sur l'écran. Les mots à afficher doivent être mis entre guillemets (''). Mettre la vitesse à 8 **(SPEED=8)** et introduire le programme suivant:



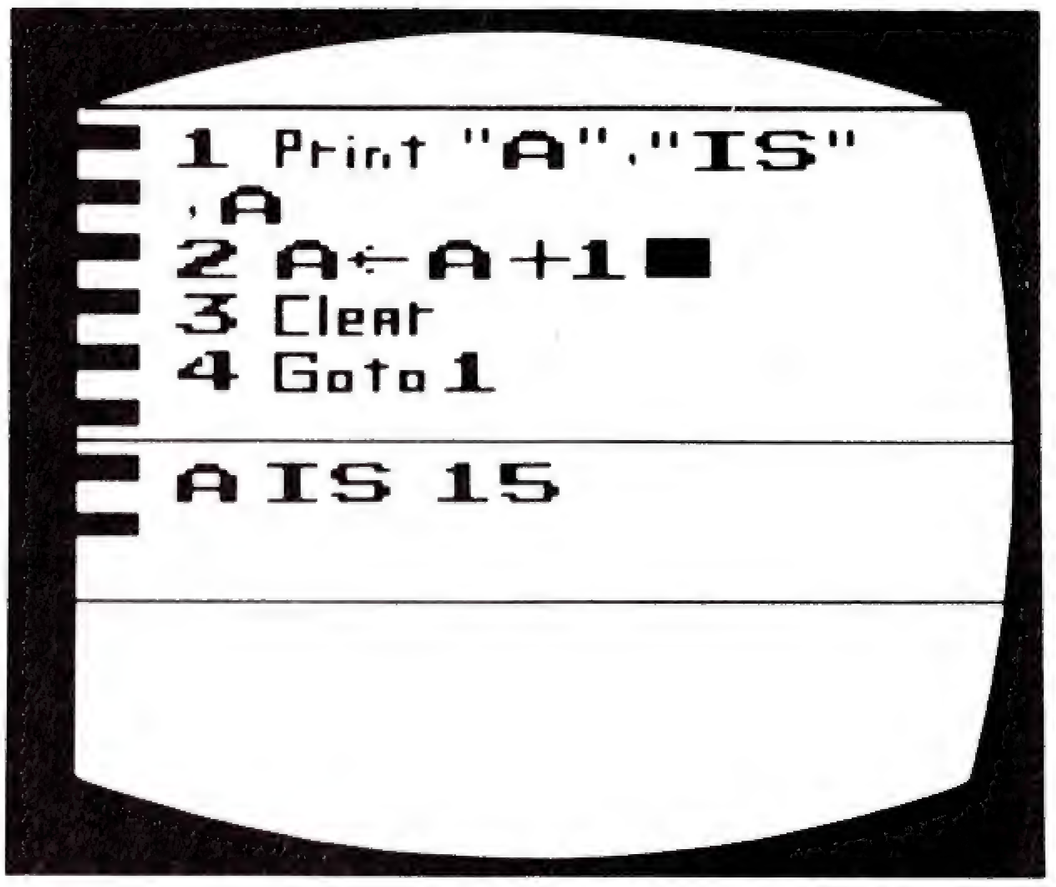
Exécuter ensuite le programme et observer la zone **OUTPUT**. Le programme **imprime** les mots introduits, mais ils sont "empilés."



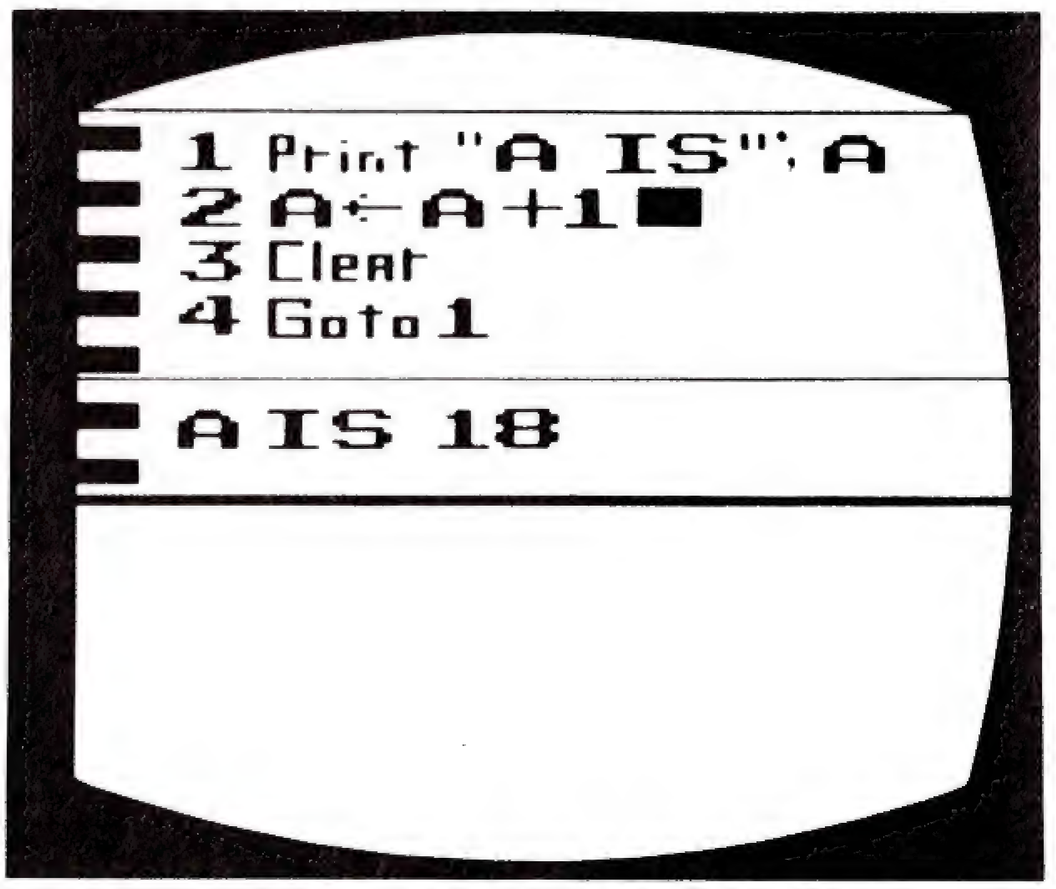
Modifier légèrement le programme en ajoutant une virgule (,) à la fin des lignes **1** et **2**. Le programme se présente alors ainsi:



La virgule (,) ajoutée à la ligne **1** indique au programme que le contenu de la ligne **2** doit être affiché dans la zone **OUTPUT** sur la même ligne que l'instruction de la ligne **1**. La virgule ajoutée à la ligne **2** indique au programme que l'instruction de la ligne **3** doit venir à la suite de la ligne **2**. On peut abréger ces instructions en modifiant le programme comme suit:



Il est possible d'abréger encore le programme:



En rédigeant le programme de cette façon, on utilise aussi moins de mémoire.

UTILISATION DE LA ZONE GRAPHICS

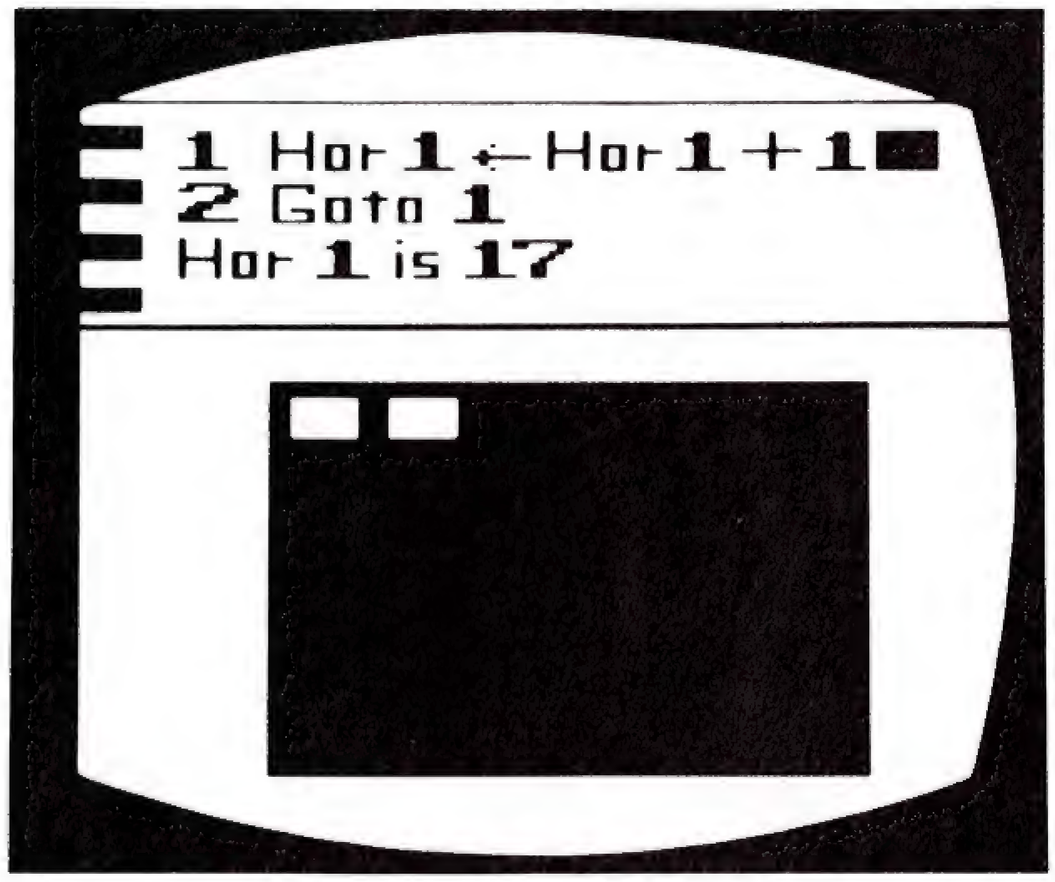
La zone **GRAPHICS** est représentée par un rectangle bleu sur l'écran du téléviseur. À première vue, on voit qu'il contient un carré rouge dans le coin supérieur gauche. En fait, ce carré rouge en recouvre un autre, blanc; ils se déplacent dans cette zone indépendamment l'un de l'autre et suivant les ordres du programme.

Pour déplacer les carrés, on fait varier leurs coordonnées. Le carré rouge est l'objet numéro 1. Son abscisse est représentée sur le clavier par **Hor 1**, son ordonnée par **Ver 1**. **Hor 2** et **Ver 2** sont les coordonnées de l'objet numéro 2, le carré blanc.

Les deux objets, ou carrés, partent du coin supérieur gauche de la zone bleue, position dans laquelle seul le carré rouge est visible. Le carré supérieur gauche représente la position zéro (0), l'origine. Lorsqu'aucune valeur n'a été affectée aux variables **Hor 1, Ver 1, Hor 2** et **Ver 2** (lorsqu'elles ne sont pas définies), leur valeur est nulle (et les carrés restent dans le coin supérieur gauche).

Lorsqu'on affecte une valeur autre que zéro à l'une des coordonnées (par exemple: **Hor 1** ← 10), l'objet en question saute à la position correspondante de la zone bleu lorsqu'on exécute le programme.

Entrer le programme suivant:

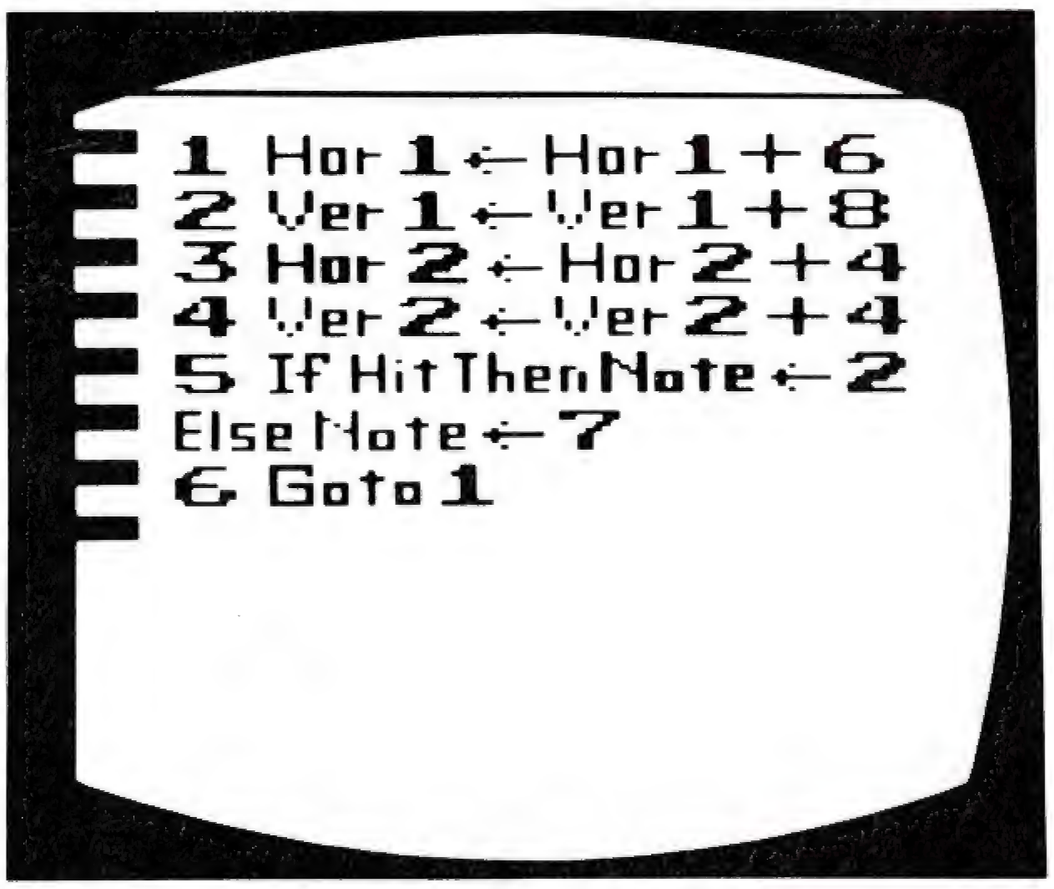


Ce programme déplace le carré rouge vers la droite, d'un espace à la fois. Lorsqu'on l'exécute, on voit le carré rouge se déplacer lentement vers la droite. Si l'on observe la zone **VARIABLES**, on voit que la valeur de **Hor 1** augmente. Elle augmente jusqu'à ce qu'elle atteigne **99**, la valeur possible la plus élevée, puis retombe à **0**.

Lorsque **Hor 1** atteint **99**, le carré rouge atteint la limite droite de la zone bleue. Il passe alors "Par-dessous" et reparaîte à la limite gauche de la zone lorsque **Hor 1** égale **0**.

Les abscisses (**Hor 1** et **Hor 2**) varient donc de 0 à 99. Les ordonnées (**Ver 1** et **Ver 2**) varient également de 0 à 99, le 0 se trouvant en haut de la zone bleue et le 99 en bas.

Entrer le programme suivant:



Faire disparaître le programme de l'écran et s'assurer que la zone **GRAPHICS** est entièrement visible. Régler la vitesse à 60 (**SPEED = 60**) et exécuter le programme. Ce programme montre une manière de déplacer les carrés dans la zone bleue. Il indique également à quo peuvent servir les fonctions **HIT** (toucher) et **ELSE** (sinon).

À la ligne **5**, le programme ordonne à l'ordinateur d'émettre la note **2** si (**IF**) les carrés se touchent (**HIT**), ce qui se produit périodiquement. (On dit que les carrés se touchent (**HIT**) lorsqu'ils ont les mêmes coordonnées.) Sinon (**ELSE**), l'ordinateur doit émettre la note **7**, ce qu'il fera jusqu'à ce que les carrés se touchent.

(Si l'on fait disparaître **ELSE NOTE ← 7** (sinon Note ← 7) de la ligne **5**, alors la note **2** est émise chaque fois que les carrés se touchent et aucune autre note n'est émise.)

UTILISATION DE LA FONCTION MOD

Mod (modulo) est un opérateur arithmétique très semblable à l'opérateur de la division (: ou ÷). Le BASIC PROGRAMMING utilise la division par nombres entiers, c'est-à-dire qu'il n'utilise que des nombres entiers sans tenir compte des restes. Par exemple, à quoi est égal 14 divisé par 5? On peut répondre 2 et 4/5, ou bien 2, reste 4. En BASIC PROGRAMMING 14 : 5 = 2. Bien qu'il y ait deux fois 5 (soit 10) dans 14 et qu'il reste 4, la réponse en BASIC est 2 parce que le BASIC n'utilise que des nombres entiers.

Que l'on divise 12 par 4 ou 13 par 4, le résultat indiqué par l'ordinateur est le même: 3 dans les deux cas puisque 12 aussi bien que 13 contiennent trois fois 4. L'ordinateur ne tient pas compte du 1 qui reste quand on divise 13 par 4.

Voyons maintenant la fonction **Mod: Mod** indique le reste de la division du premier nombre par le second. Par exemple, **14 Mod 5** (14 modulo-5) est égal à **4**. Cinq est contenu deux fois dans 14 (ce qui fait 10), le reste est 4. **Mod** indique le reste et est donc égal à 4.

À quoi est égal **13 Mod 4**? La réponse est 1, puisqu'il reste **1** lorsqu'on divise **13** par **4** (soit 12 par 4). Et **12 Mod 4**? Dans ce cas, **Mod** est égal à **0** puisque 12 est divisible par 4 et qu'il ne reste rien (soit **0**).

Normalement, la division par 0 n'est pas définie, mathématiquement parlant. Dans le BASIC PROGRAMMING, le résultat de la division par 0 est 0, si bien que 5 : 0 = 0.

ORDRE DE PRIORITÉ DES OPÉRATEURS

Dans une équation, les opérations arithmétiques (+ , –, X, : ou ÷, Mod, etc.) sont effectuées dans un certain ordre de priorité. Le BASIC PROGRAMMING utilise l'ordre de priorité suivant:

1. X : (priorité la plus élevée)
2. + –
3. Mod
4. =
5. ← (priorité la plus faible)

UTILISATION DES PARENTHÈSES

Les parenthèses (mode vert, côté gauche du clavier) servent à indiquer que les nombres qu'elles contiennent doivent avoir priorité lorsqu'on calcule une expression.

Par exemple, pour calculer l'expression suivante:

A 5 + 3 X 2 Mod 7

On calcule d'abord: 3 X 2 = 6
ensuite: 5 + 6 = 11
et finalement: 11 Mod 7 = 4
A = 4

Toutefois, si l'on ajoute des parenthèses, le calcul se fait dans un ordre différent et la valeur obtenue pour A n'est plus la même.

A (5 + 3) X 2 Mod 7

On calcule d'abord: 5 + 3 = 8
ensuite: 8 X 2 = 16
et finalement: 16 Mod 7 = 2
A = 2

EXEMPLES DE PROGRAMMES

On trouvera ci-après quelques exemples de programmes qui permettront de se rendre compte des multiples possibilités offertes. Ne pas oublieer d'observer attentivement comment un ordre donné (**IF, THEN, HIT, ELSE, PRINT, KEY, etc.**) affecte le programme. Ne pas oublier non plus que l'effet de l'ordre **KEY** sur le bon déroulement du programme pourra dépendre de la valeur entrée par le côté droit du clavier.

Lorsqu'un programme donné paraît confus, il faut l'interrompre, revenir au début, et l'exécuter pas à (**STEP**). En observant le programme dans la zone **STACK**, on voit comment l'ordinateur exécute chaque étape, ce qui permet d'en comprendre le pourquoi et le comment.

Après avoir exécuté ces programmes et s'être familiarisé avec les notions de base de la programmation des ordinateurs en général, on pourra songer à écrire quelques programmes soi-même.

HIT (toucher)

```
1 Clear PrintHit
2 Hor 2←Hor 2+1 Mod
3 3
3 Goto 1
```

MUSIQUE

```
1 Note←If Note>4
Then Note—11 Else
Note+1
2 Note←Note+2
3 Note←Note—2
4 Goto 1
```

```
1 A←Key
2 If A>0 Then Note←A
—1
3 Goto 1■
```

```
1 Note←Note+Key+
7
2 Goto 1■
```

```
1 A←A+1
2 Note←A
3 If (A Mod 2)=0 Then
Note←0 Else Note←4
4 Goto 1■
```

PROGRAMME D'HORLOGE

Pour l'exécution du programme d'horloge, n'amener que la zone OUTPUT sur l'écran. Régler la vitesse **(SPEED)** à 30 ou 60.

```
1 Clear
2 A←A+1 Mod 60
3 If A=0 Then B←B+1
Mod 60
4 Print B.A
5 Goto 1
```

JEU DE PONG®
(Balle et raquette)

```
1 Hor 1←Hor 1+Key
2 Hor 2←Hor 2+8
3 Ver 2←Ver 2-3
4 If Hit Then Ver 2←93
. Note←7
5 Goto 1
```

PONG® GAME
sans bruitage

```
1 Hor 2←Hor 2+Key
2 If Ver 1>90 Then
Ver 1←88
3 If Hit Then Ver 1←3
4 Ver 1←Ver 1+If
Ver 1 Mod 2 Then 8 Else
92
5 Hor 1←Hor 1+7
6 Goto 1
```

1 VIDEO-SPIEL

BASIC PROGRAMMING

EINLEITUNG

BASIC PROGRAMMING ist ein Lehrmittel, das dazu bestimmt ist, die wesentlichen Phasen der Computerprogrammierung zu lehren. **BASIC** ist ein Akronym für **B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. Diese Computersprache wurde geschaffen, um das "Schreiben"-Lernen von Computerprogrammen zu erleichtern.

Computerprogramme sind ganz einfach eine Reihe von Befehlen. Das Programm steuert den Informationsfluß innerhalb des Computers. **BASIC PROGRAMMING** gestattet Ihnen, dem Video Computer System™ die Befehle zu erteilen, die es benötigt, um einige einfache Aufgaben auszuführen.

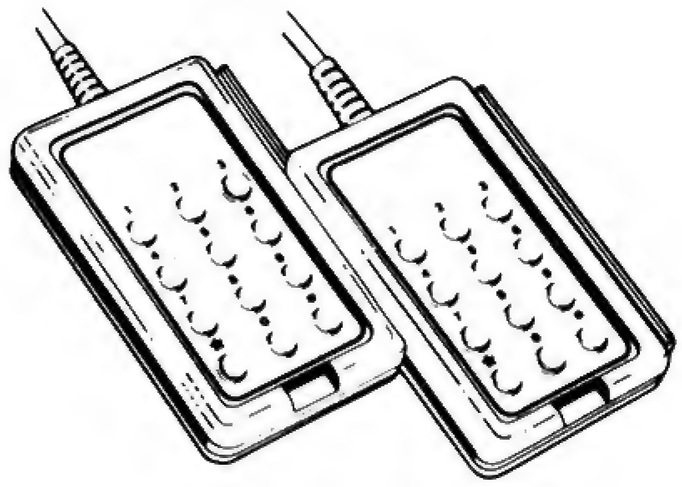
Vergessen Sie nicht, daß **BASIC PROGRAMMING** im Vergleich mit komplizierteren Computersystemen über eine begrenzte Speicherkapazität verfügt. Es ist jedoch ein ausgezeichnetes Lehrmittel, um die Grundlagen der Computer-Programmierung zu erlernen.

HINWEIS: Schalten Sie vor dem Einschieben oder Herausnehmen eines ATARI® Game Programs™ den Konsolenschalter (**POWER**) stets aus (**OFF**). Dadurch werden die elektronischen Bestandteile geschützt und die Nutzlebensdauer Ihres ATARI® Video Computer Systems™ wird verlängert.

Dieses Game Program™ könnte bei manchen Geräten ein "Rollen" des Fernsehbildschirmes verursachen. Dieser Vorgang kann durch den VERTIKALEN BILDFANG (VERTICAL HOLD) ausgeglichen werden.

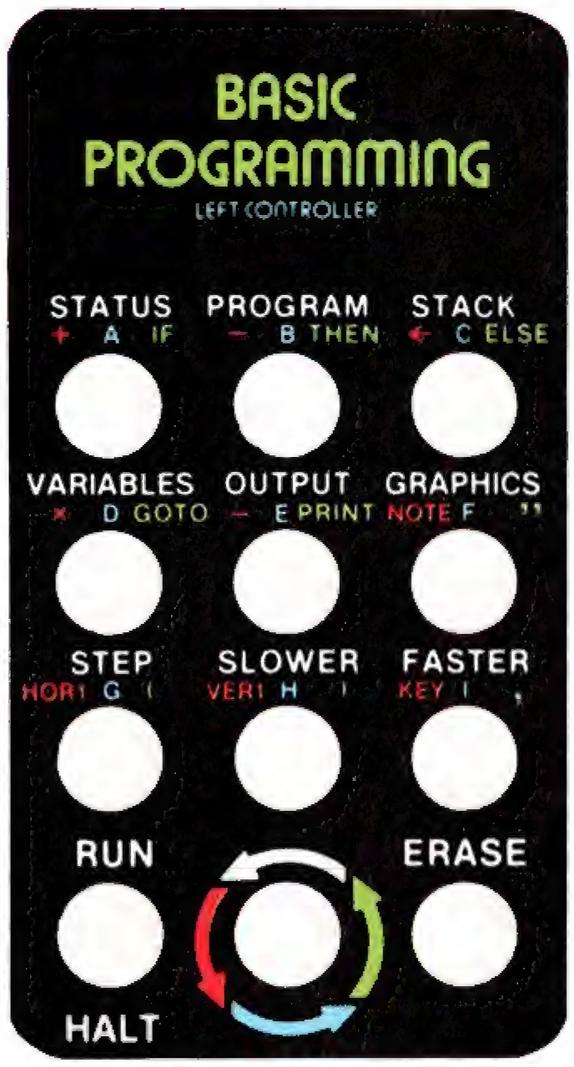
TASTATURSTEUERUNG

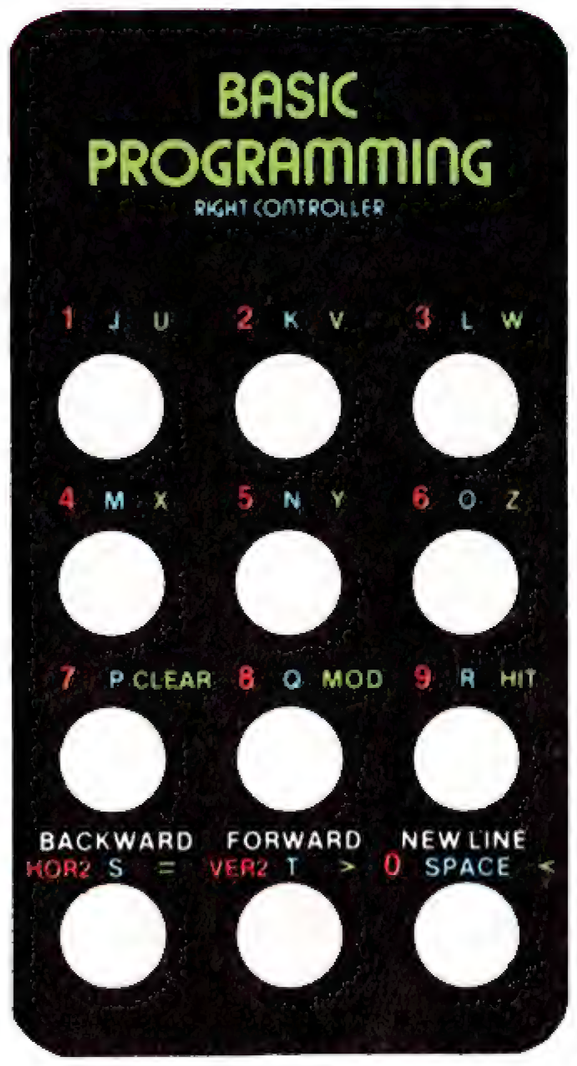
Mit diesem ATARI® Game Program™ benützen Sie die Tastatursteuerungen. Überzeugen Sie sich, daß das jeweilige Kabel fest in der Steuerungs-(**CONTROLLER**)-Buchse an der Rückwand Ihres ATARI® Video Computer Systems™ steckt. *Details finden Sie in Abschnitt 3 Ihrer Video Computer System™ Gebrauchsanweisung.*



Um die beiden Steuerungen zu verbinden, schieben Sie die Feder der linken Steuerung in die Nut der rechten Steuerung. Die beiden, zusammen eingerasteten Steuerungen bilden eine 24-Tasten-Tastatur, die zur Eingabe Ihres Programmes und zur Steuerung der Darstellung an Ihrem Fernsehbildschirm benützt wird.

Nehmen Sie die Tastatursteuerungs-Aufkleber aus dem Umschlag heraus. Kleben Sie den **LEFT** (LINKS) markierten Aufkleber über





die Tastatur, die in die **LINKE STEUERUNGSBUCHSE** (LEFT CONTROLLER) an der Rückwand der Computer-Konsole eingesetzt ist. Kleben Sie den Aufkleber mit **RIGHT** (RECHTS) über die Tastatur, die in die **RECHTE STEUERUNGS-BUCHSE** (RIGHT CONTROLLER) an der Rückwand Ihrer Com-puterkonsole eingesetzt ist.

ANZEIGENBEREICHE

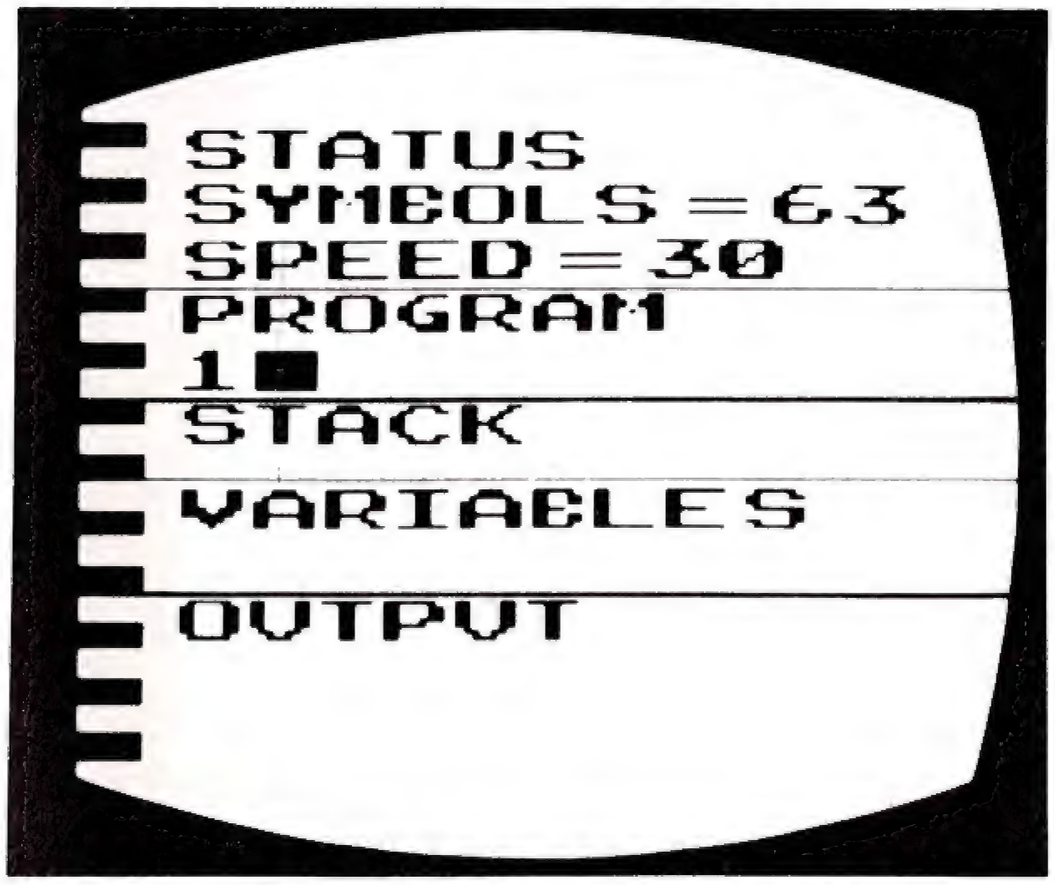
Die Anzeige ist in sechs Bereiche aufgeteilt:

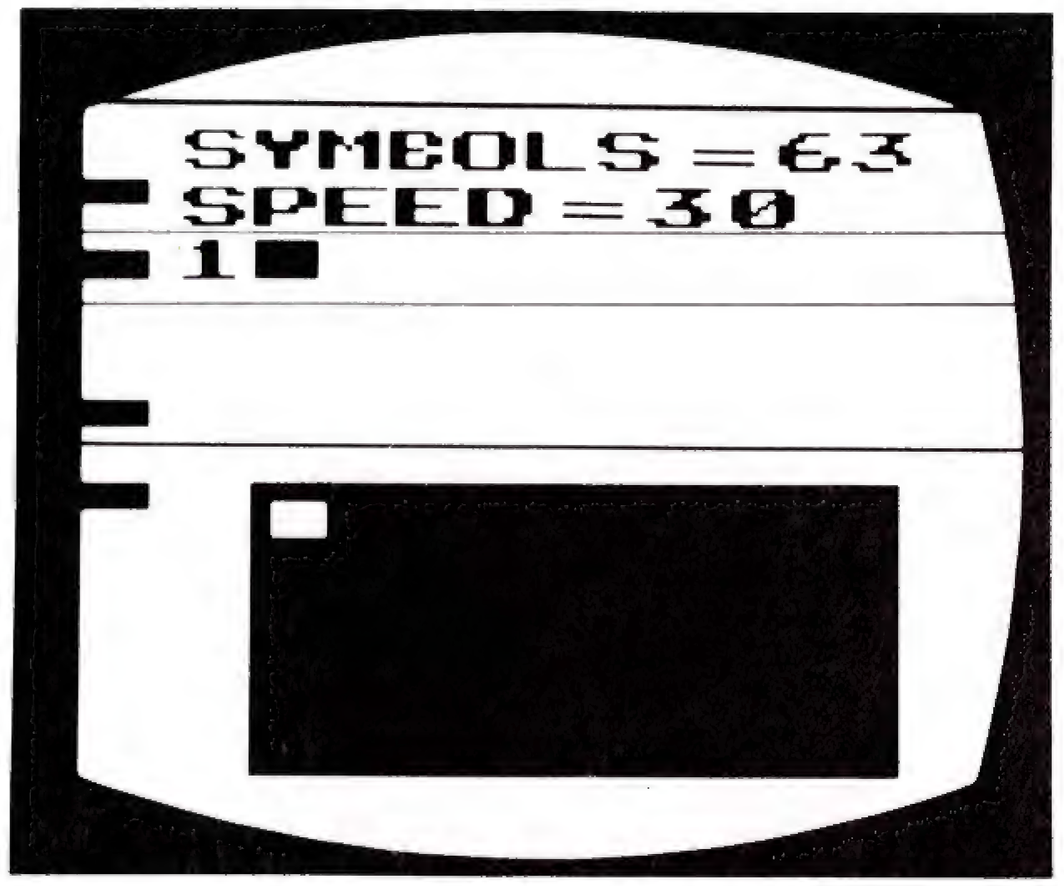
1. Der **PROGRAM** Bereich wird benützt, um dem Computer Befehle zu übermitteln.
2. Der **STACK** Bereich gibt die vorläufigen Resultate des Programmes, während der Computer es fährt.
3. Der **VARIABLES** Bereich gibt den Wert jeder Variablen des Programmes, während es gefahren wird.
4. Der **OUTPUT** Bereich zeigt die generierte Ausgabe, während das Programm gefahren wird.
5. Der **STATUS** Bereich zeigt auch, wieviel Speicher zu jeder Zeit für das Programm zur Verfügung steht. Dieser Bereich zeigt die Schnelligkeit des Programmablaufes an.
6. Der **GRAPHICS** Bereich enthält zwei farbige Quadrate, die unter Steuerung des Programmablaufes bewegt werden können.

Bevor Sie mit Ihrem Programm beginnen, bringen Sie den **linken Schwierigkeitsschalter** (LEFT DIFFICULTY) in die Stellung **b**. So sehen Sie, wo sich jeder Bereich an Ihrer Anzeige befindet. Wenn der **linke Schwierigkeitsschalter**

in der Stellung **a** ist, verschwinden die Überschriften für jeden Bereich von der Anzeige, und der GRAPHICS Bereich wird sichtbar gemacht.

In diesem Game Program™ hat der **rechte Schwierigkeitsschalter** (RIGHT DIFFICULTY) keine Funktion.





DIE POSITIONSANZEIGE

Bringen Sie den **LEFT DIFFICULTY** Schalter in die Stellung **b** und schalten Sie die Konsole **AUS** (OFF), dann wieder **ON** (EIN). Im **PROGRAM** Bereich befindet sich ein weißes Rechteck. Dies ist die Positionsanzeige. Die *Umschalttaste* befindet sich in der Mitte der unteren Reihe der linken Steuerung. Drücken Sie diese Taste vier Mal nieder. Die Farbe der Positionsanzeige wechselt darauf von weiß auf rot, von rot auf blau, von blau auf grün, und schließlich von grün wieder auf weiß.

Die Positionsanzeige dient dazu, das Programm einzugeben. Jede der vier Farben an der Umschalttaste entspricht den farbigen Befehlen oder Eingaben der Tastatur. Die weiße Betriebsart wird benutzt, um dem Computer Befehle zu erteilen, während die anderen Betriebsarten dazu dienen, Symbole ins Programm einzufügen.

BEWEGUNG DER POSITIONSANZEIGE VON BEREICH ZU BEREICH

Überzeugen Sie sich, daß sich der **LEFT DIFFICULTY** Schalter in der Stellung **b** befindet, und schalten Sie die Konsole **AUS** (OFF), dann wieder **EIN** (ON). Drücken Sie die **FORWARD** Taste, und die Positionsanzeige bewegt sich von **PROGRAM** zu **STACK**. Drücken Sie die **FORWARD** Taste wieder, und die Positionsanzeige bewegt sich von **STACK** zu **VARIABLES**. Wenn Sie die Taste nochmals drücken, bewegt sich die Positionsanzeige von **VARIABLES** zu **OUTPUT**.

Wenn Sie die **BACKWARD** Taste betätigen, können Sie die Positionsanzeige in den **PROGRAM** Bereich zurückbringen. Die Tasten **FORWARD** und **BACKWARD** können jeweils einmal für jeden Bereich gedrückt, oder niedergehalten werden.

Jetzt bringen Sie den **LEFT DIFFICULTY** Schalter in die Stellung **a**. Die Überschriften in jedem Bereich verschwinden und ein Teil des **GRAPHICS** Bereiches erscheint. Bewegen Sie die Positionsanzeige wieder schrittweise durch jeden Bereich. Sie werden merken, daß die Positionsanzeige sich nicht in den **GRAPHICS** Bereich bewegt.

ENTFERNUNG DER BEREICHE VON DER ANZEIGE

Bringen Sie den **LEFT DIFFICULTY** Schalter wieder in die Stellung **b** und schalten Sie die Konsole **AUS** (OFF), und dann **EIN** (ON). An der linken Seite der Tastatur finden Sie eine Reihe von Befehlen (weiße Betriebsart), die jedem Bereich entsprechen: **STATUS, PROGRAM, STACK, VARIABLES, OUTPUT**, und **GRAPHICS**. Fangen wir mal mit dem **STATUS** Bereich an. Drücken Sie die **STATUS** Taste einmal, und der **STATUS** Bereich verschwindet vom Bildschirm; der **PROGRAM** Bereich bewegt sich auf der Anzeige nach oben.

Drücken Sie die **PROGRAM** Taste, verschwindet der **PROGRAM** Bereich, um dem **STACK** Bereich Platz zu machen, der sich nach oben an die Anzeige bewegt. Wenn Sie die **STACK** Taste drücken, verschwindet der **STACK** Bereich, und der **VARIABLES** Bereich bewegt sich nach oben zur Anzeige. Wenn Sie die **VARIABLES** Taste drücken, verschwindet der **VARIABLES** Bereich, und der **OUTPUT** Bereich kommt nach oben an die Anzeige.

Sie entfernen den **OUTPUT** Bereich, indem Sie die **OUTPUT** Taste drükken. Dies bewirkt ferner, daß der **GRAPHICS** Bereich an der Anzeige ganz sichtbar wird. Drücken Sie die **GRAPHICS** Taste, um den **GRAPHICS** Bereich zu entfernen. Ihre Anzeige sollte keine Bereiche zeigen.

Jetzt drücken Sie die **STACK** Taste. Der **STACK** Bereich erscheint von neuem an Ihrer Anzeige. Entfernen Sie den **STACK** Bereich, und bringen Sie die Bereiche **OUTPUT** und **VARIABLES** nach oben. Sie

können jeden Bereich darstellen oder entfernen, wenn sich der **LEFT DIFFICULTY** Schalter in der Stellung **a** oder **b** befindet.

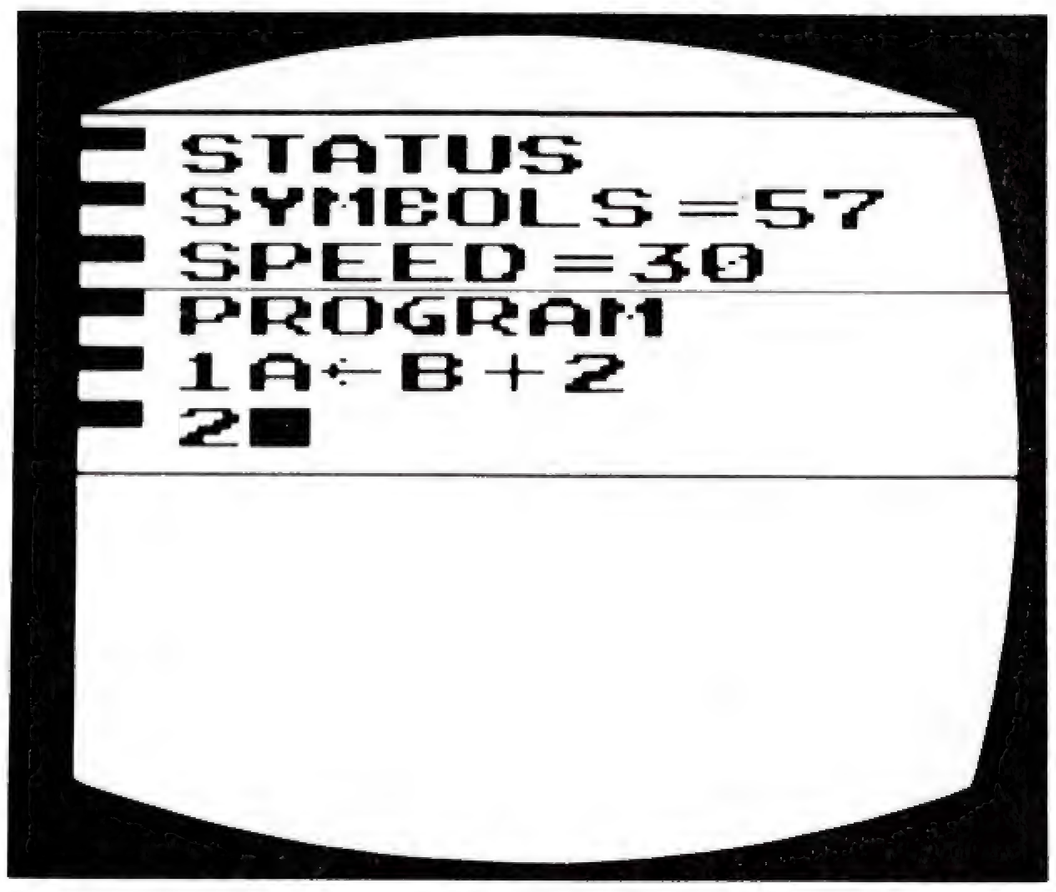
Beachten Sie bitte, daß die Darstellung oder Nicht-Darstellung der verschiedenen Bereiche den Programmablauf nicht beeinflußt.

EIN PROGRAMM FAHREN

Fangen wir mit einem einfachen Programm an. Überzeugen Sie sich, daß sich der **LEFT DIFFICULTY** Schalter in der Stellung **b** befindet, und schalten Sie die Konsole **AUS** (OFF), und dann wieder **EIN** (ON). Ferner können Sie ein Programm löschen und sämtliche Werte rücksetzen, indem Sie den **Spielrücksetzschalter** (GAME RESET) drücken. Mit dem Drücken des **GAME RESET** Schalters werden sämtliche Werte gelöscht und das Programm ohne Programmlöschung zum Anfang zurückgesetzt.)

Entfernen Sie die Bereiche **STACK, VARIABLES, OUTPUT** und **GRAPHICS** von der Anzeige. Die Positionsanzeige ist dann in der weißen Betriebsart, und rechts von der Nummer **1** im **PROGRAM** Bereich. Ändern Sie die Positionsanzeige auf blau, und drücken Sie die Taste **A**. Der Buchstabe **A** erscheint an der Anzeige neben der **1**. (Jede Zeile ist numeriert, wodurch zu erkennen ist, wo eine Zeile endet und die nächste beginnt). Jetzt ändern Sie die Positionsanzeigenfarbe auf rot, und drücken die ← Taste. Ein kleiner Pfeil erscheint neben dem **A**. Gehen Sie nun zu blau zurück, und geben Sie **B** ein. Wenn Sie die Positionsanzeige auf rot ändern, geben Sie + ein, und die Nummer **2**. Jetzt gehen Sie zurück zur weißen Betriebsart, und geben **New Line** ein. Sie müssen stets in der weißen Betriebsart sein, um eine neue Zeile in Ihrem Programm zu beginnen.

Ihre Anzeige sollte so aussehen:

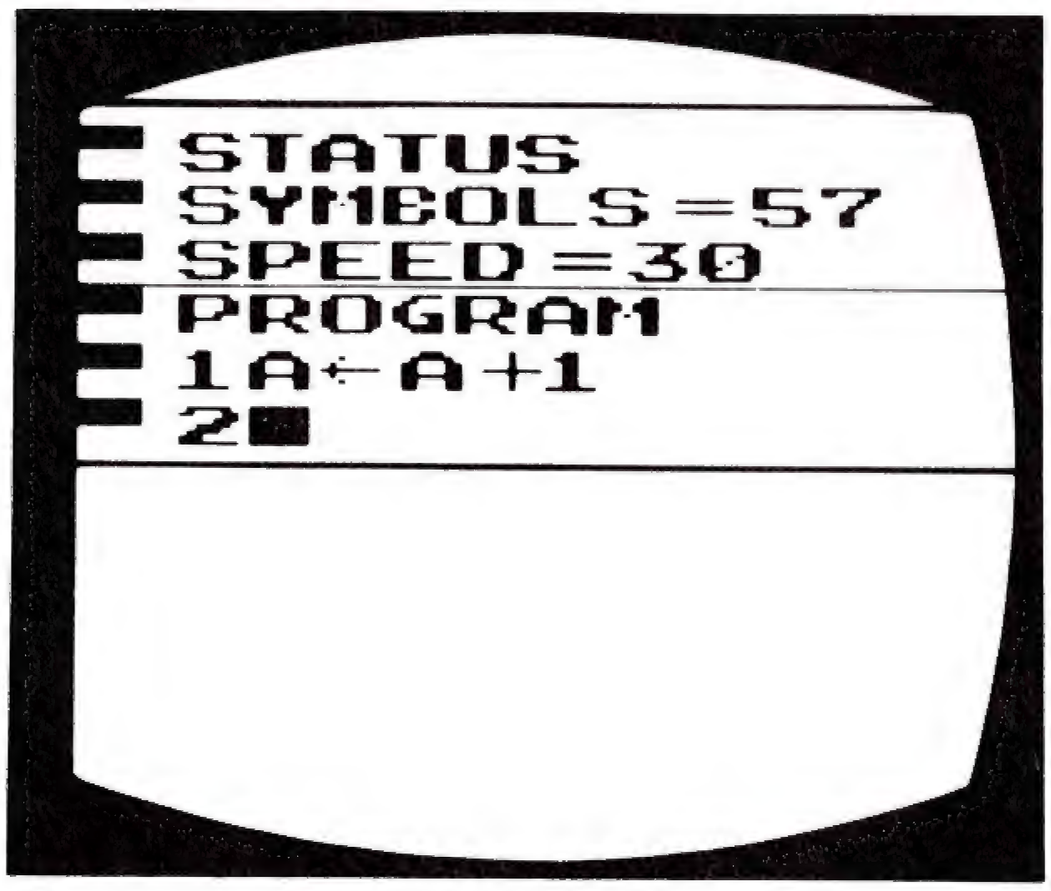


Wenn Sie bei der Eingabe Ihres Programmes einen Fehler machen, können Sie diesen mit Hilfe der **ERASE** Taste löschen. Beachten Sie, daß die **ERASE** Taste nicht farbcodiert ist. Sie kann benützt werden, wenn sich die Umschalttaste in irgendeiner Farbbetriebsart befindet.

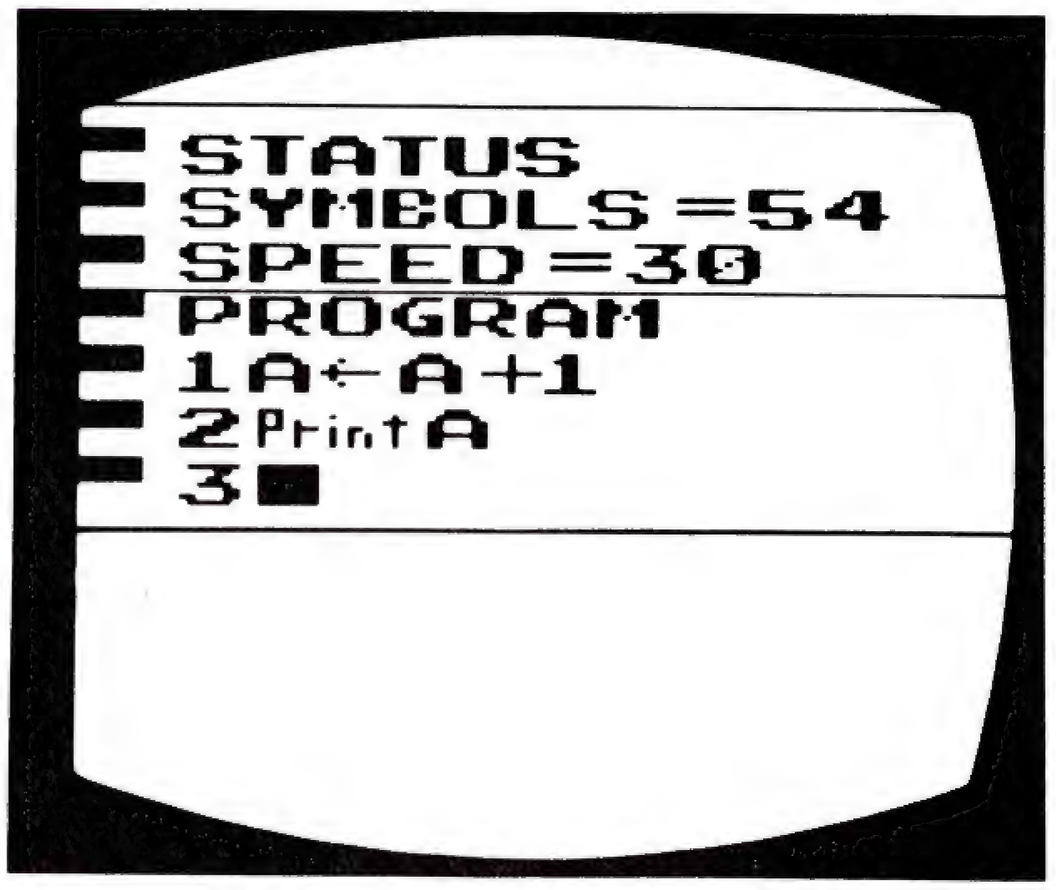
Wir wollen nun mit der Zeile, die Sie gerade in Ihr Programm eingegeben haben, eine Übung durchführen. Drücken Sie die **ERASE** Taste einmal. Die Positionsanzeige bewegt sich von Zeile **2** nach rechts direkt neben die **2** auf Zeile **1**. Drücken Sie

wieder die **ERASE** Taste, und die **2** wird aus dem Programm gelöscht. Jetzt geben Sie **1** in der roten Betriebsart ein. Ändern Sie die Positionsanzeige auf die weiße Betriebsart und benützen Sie die **BACKWARD** Taste, um die Positionsanzeige zu bewegen, bis sie rechts direkt neben dem **B** im Programm ist. Drücken Sie die **ERASE** Taste und die **B** wird vom Programm entfernt. Ersetzen Sie diesen Buchstaben jetzt mit **A** (blaue Betriebsart). In der weißen Betriebsart benützen Sie die **FORWARD** Taste, um zum Ende der Zeile **1** zu kommen und **New Line** einzugeben.

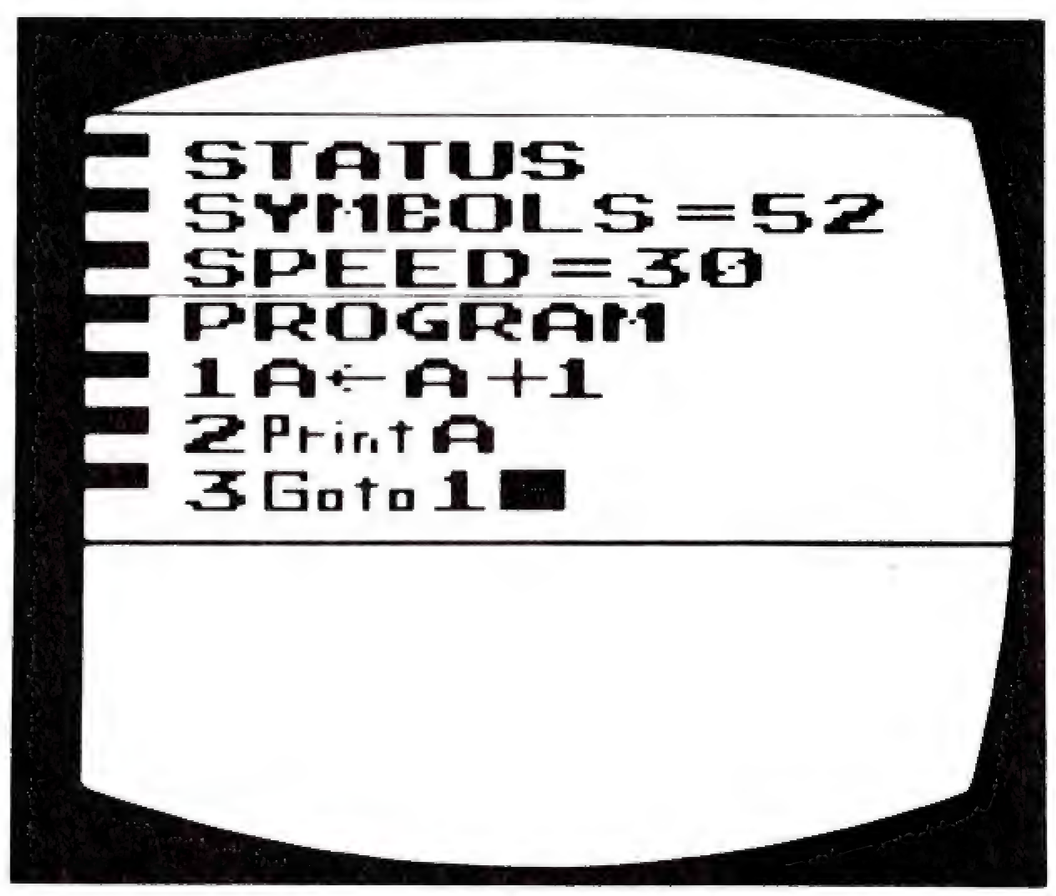
Vergessen Sie nicht, daß die Positionsanzeige sich nicht am auszulöschenden Symbol, sondern direkt rechts davon befinden soll. Ihre Bildschirmdarstellung wird jetzt so aussehen:



Die Positionsanzeige befindet sich rechts direkt neben der **2** in Zeile **2**. Geben Sie jetzt **PRINT** in der grünen Betriebsart ein. Danach benützen Sie die blaue Betriebsart, um **A** einzugeben, und zu einer neuen Zeile zu gehen. Ihre Anzeige sieht dann so aus:



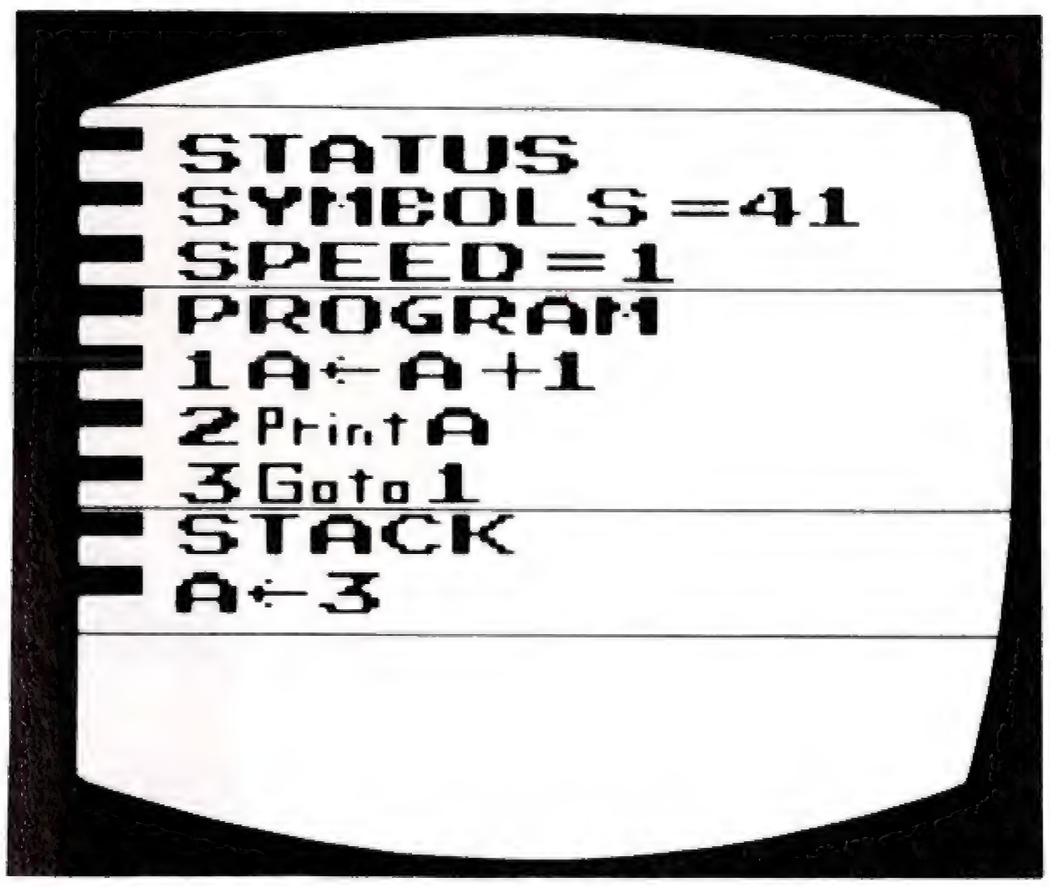
Jetzt geben Sie in der grünen Betriebsart mit der Positionsanzeige **GOTO** ein und dann, in der roten Betriebsart, geben Sie **1** ein. Bevor wir etwas anderes machen, wollen wir die Bildschirmdarstellung nochmals überprüfen.



Beachten Sie, daß der **STATUS** Bereich zeigt, daß wir 53 "Bytes" oder Symbole von übrigem Speicher haben. **SPEED** (Geschwindigkeit) ist auf 30 eingestellt (**SPEED = 30**). Bringen Sie die Positionsanzeige in

die weiße Betriebsart und drücken Sie die **SLOWER** Taste. Jedes Mal, wenn diese Taste gedrückt wird, verringert sich die Geschwindigkeit. Drücken Sie die **FASTER** Taste und die Geschwindigkeit erhöht sich.

Bevor wir mit dem Programm beginnen, wird **SPEED = 1** eingegeben. Dann wird die **STACK** Taste gedrückt. Drücken Sie die **RUN/HALT** Taste zweimal, und das Programm beginnt. Se können beobachten, wie der Computer jeden Teil des Programms abarbeitet.



Um das Programm zu stoppen, drücken Sie **RUN/HALT**. Wie oben bemerkt, werden durch Drücken des **GAME RESET** Schalters sämtliche Werte in Ihrem Programm gelöscht (bis auf das eigentliche Programm) und das Programm zum Anfang zurückgesetzt.

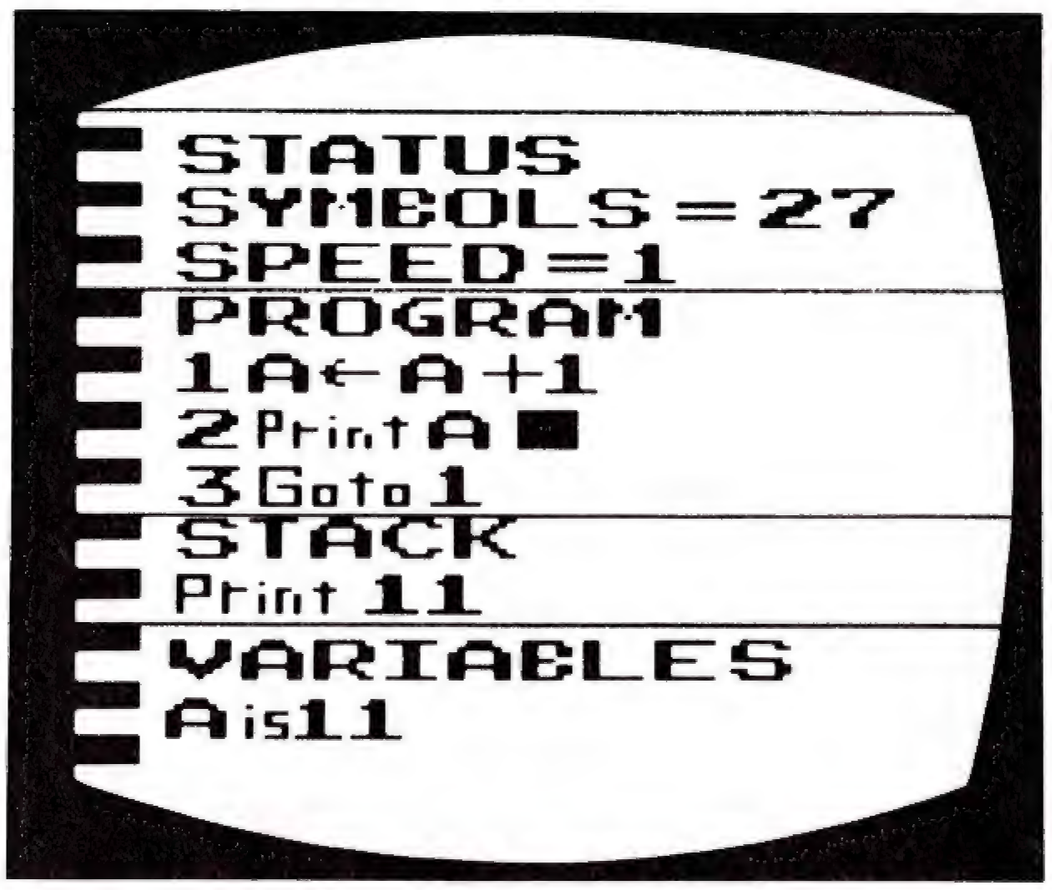
Wir wollen das Programm Schritt für Schritt analysieren, wie der Computer es durchläuft. Ändern Sie **SPEED** auf **60**. Mit der Positionsanzeige in der weißen Betriebsart drücken Sie die **STEP** Taste. So erhalten Sie jeden Teil des Programms Schritt für Schritt, jedes Mal, wenn die Taste gedrückt wird.

In Zeile **1** steht im Programm **A ←— A + 1**. Der Computer liest dies als **A** "wird zu" **A + 1**. Achten Sie auf den **STACK** Bereich, während Sie Schritt für Schritt durch Zeile 1 gehen. In Zeile **2** haben Sie dem Computer befohlen, A zu drucken, **PRINT A**. Das heißt, daß Sie vom Computer verlangen, daß er den Wert von A druckt (wie durch Zeile 1 bestimmt wird) und zwar im **OUTPUT** Bereich.

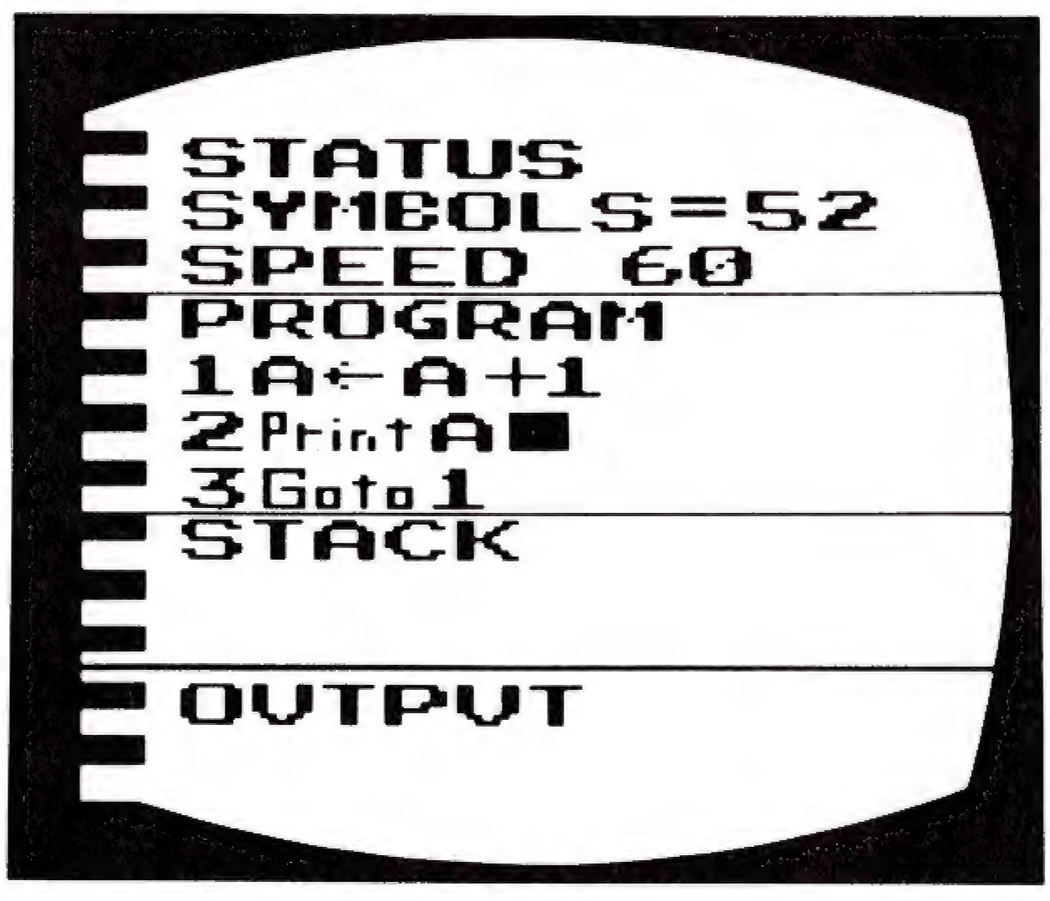
Wir werden später sehen, wie das funktioniert. Zeile 3 sagt dem Computer, daß er auf 1 gehen soll: **GOTO 1**. Der Computer soll zu Zeile 1 zurückgehen und einen neuen Wert für **A** finden. Jedes Mal, wenn das Programm durchlaufen wird, findet er einen neuen Wert für **A**, druckt diesen Wert, und kehrt zu Zeile **1** zurück. Der Computer fährt in dieser Weise fort, das Programm zu durchlaufen, bis aller "Speicher" aufgebraucht ist. Der übrige Speicher wird im **SYMBOLS** Bereich des **STATUS** Bereiches gezeigt.

Bringen Sie den **VARIABLES** Bereich auf den Bildschirm. Ändern Sie die Geschwindigkeit (**SPEED**) auf **1** und löschen Sie die Werte des Programmes, indem Sie den **GAME RESET** Schalter drücken. Drücken Sie die **RUN/HALT** Taste und beobachten Sie, wie der Computer das Programm durchläuft. Der Computer zeigt Ihnen den laufenden

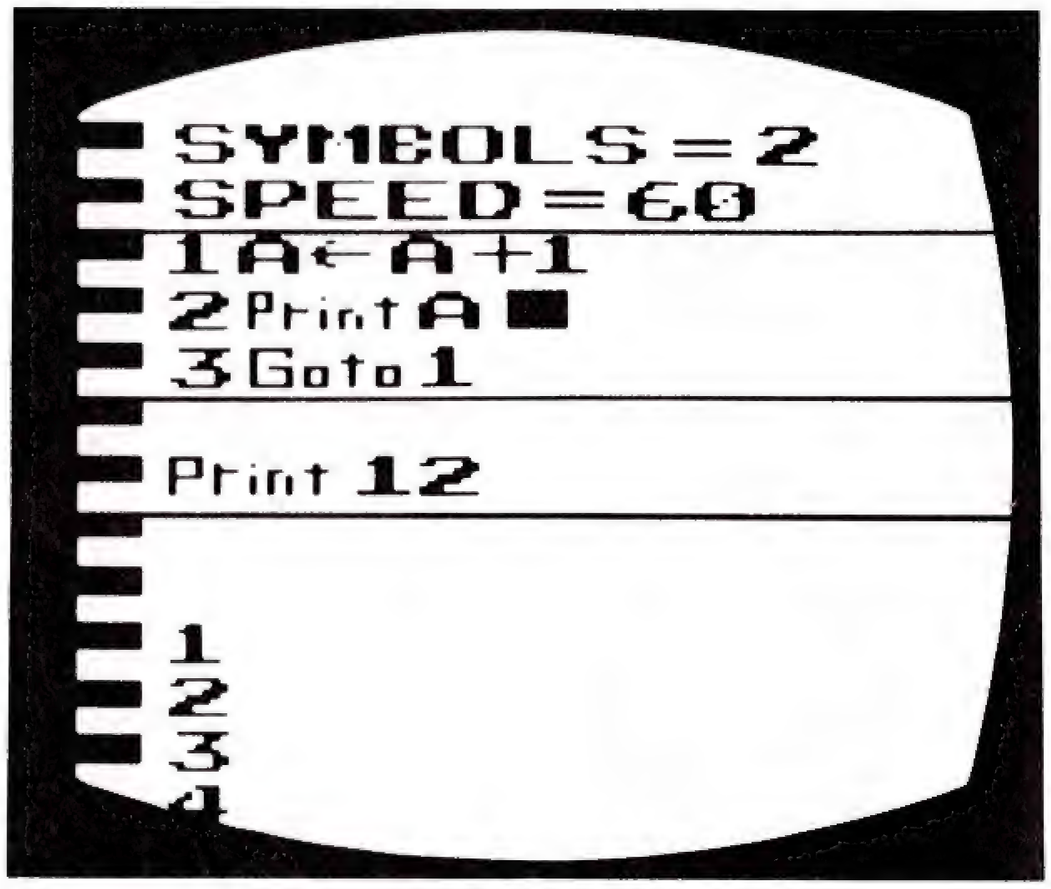
Wert von **A** während jedes Schritts Ihres Programmes. Ihre Bildschirmanzeige sieht so aus:



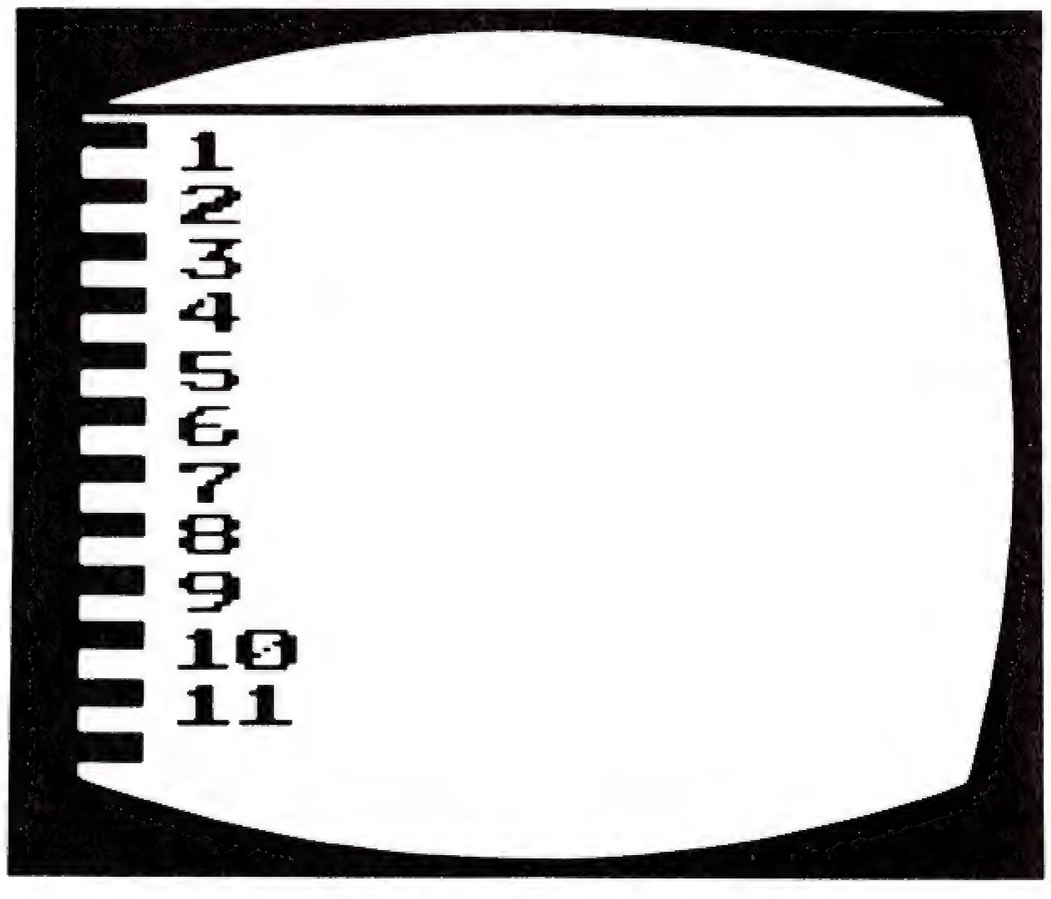
Stoppen Sie das Programm und löschen Sie die Werte (**GAME RESET**). Entfernen Sie den **VARIABLES** Bereich und bringen Sie den **OUTPUT** Bereich auf den Bildschirm. Ändern Sie die Geschwindigkeit (**SPEED**) auf **60**. Die Anzeige sieht dann so aus:



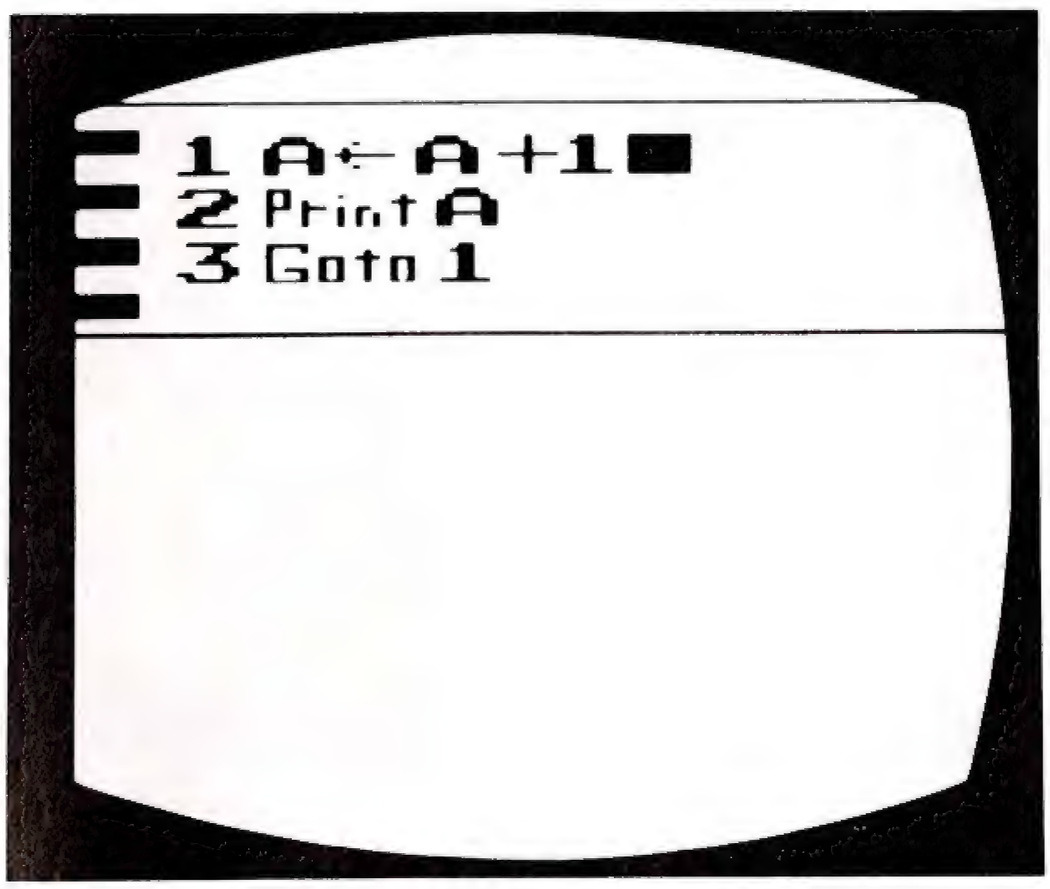
Drücken Sie die **RUN/HALT** Taste und starten Sie das Programm. In Zeile 2 wird dem Computer befohlen, **A** zu drucken: **PRINT A**. Wenn der Computer zu diesem Befehl kommt, druckt er den laufenden Wert von **A** im **OUTPUT** Bereich. Achten Sie auf den **SYMBOLS** Bereich des **STATUS** Bereiches. Obwohl **1** im **OUTPUT** Bereich Ihrer Anzeige dargestellt wird, druckt der Computer noch die sich ändernden Werte von **A**. Bringen Sie den **LINKEN DIFFICULTY** Schalter in die Stellung **a**. Die Anzeige sieht nun etwa so aus:



Wie können wir auf der Anzeige mehr aus dem **OUTPUT** Bereich zeigen? Entfernen Sie die **STATUS, PROGRAM** und **STACK** Bereiche von der Anzeige. Dann sieht sie so aus:



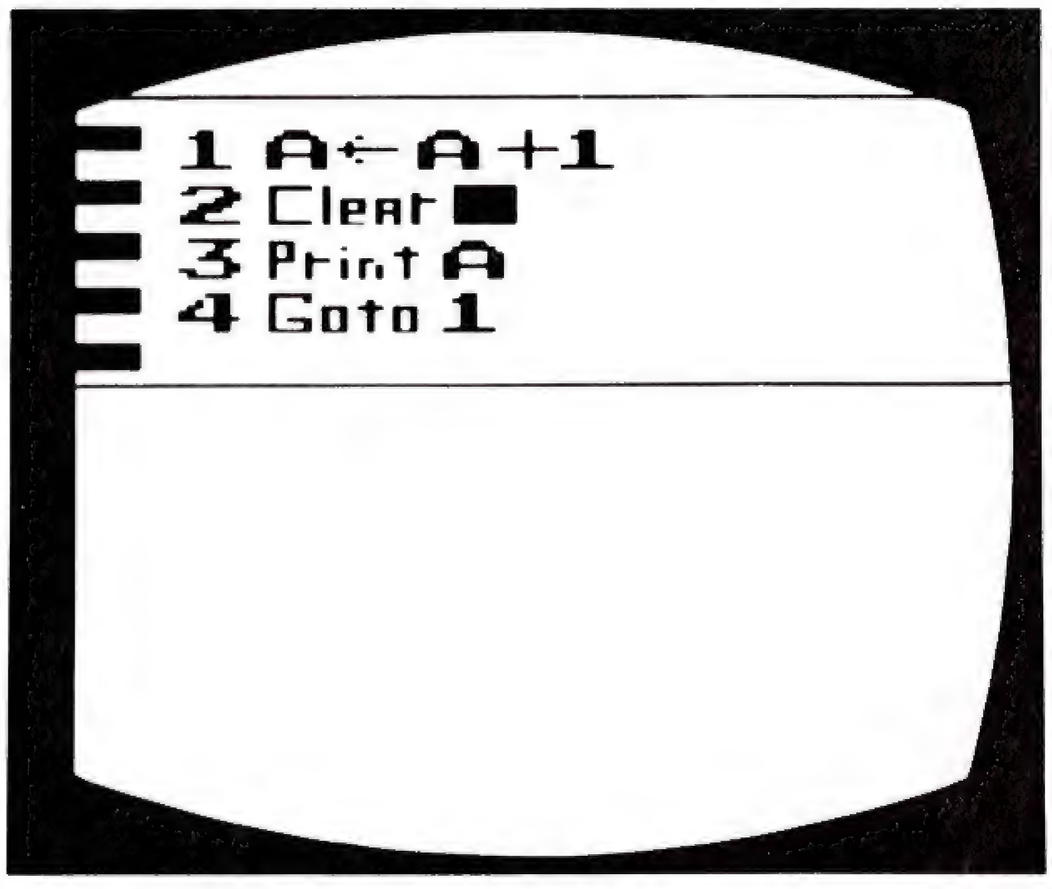
Angenommen, wir wollen den **OUTPUT** Bereich des Programmes nicht überladen: Stoppen Sie das Programm und löschen Sie die Werte mit dem **GAME RESET** Schalter. Entfernen Sie den **OUTPUT** Bereich vom Bildschirm und bringen Sie den **PROGRAM** Bereich auf die Anzeige. Benützen Sie die **FORWARD** Taste, um die Positionsanzeige zum Ende der Zeile **1** zu bringen, so daß Ihre Anzeige so aussieht:



Jetzt drücken Sie die **NEW LINE** Taste. Hier werden wir einen neuen Befehl einsetzen. Geben Sie **CLEAR** (in der grünen Betriebsart) ein. Ihre Anzeige sieht so aus:

Bringen Sie **STACK, VARIABLES** und **OUTPUT** auf den Bildschirm. Belassen Sie den **LEFT DIFFICULTY** Schalter in der Stellung **a** und beginnen Sie das Programm. Wenn das Programm Zeile **2 CLEAR** passiert, löscht es den Wert von **A** im **OUTPUT** Bereich und druckt dann den nächsten Wert von **A** in Zeile **3**.

BASIC PROGRAMMING kann nur mit zweistelligen Zahlen arbeiten;



wenn es daher 99 erreicht, beginnt es von neuem: Der Wert von A wird zu 0.

VERWENDUNG DER NOTENFUNKTION

Jedes Mal, wenn das Programm eine Nummer unter **NOTE** (rote Betriebsart) abspeichert, hört man vom Lautsprecher des Fernsehgeräts eine Note der Tonleiter.

Hier ein paar Demonstrationsbeispiele: Haben Sie irgendwelche Programme in Ihrem Video Computer System™, drücken Sie den **GAME SELECT** Schalter an der Konsole.

Geben Sie foldendes ein:

```
1    Note ←— Note + 1
2    Goto 1
```

Beginnen Sie jetzt mit dem Programm. Lassen Sie es langsamer ablaufen und beobachten Sie, was in dem **STACK** Bereich vor sich geht. Beachten Sie außerdem, daß das Programm den laufenden Wert der **NOTE** im **VARIABLES** Bereich druckt.

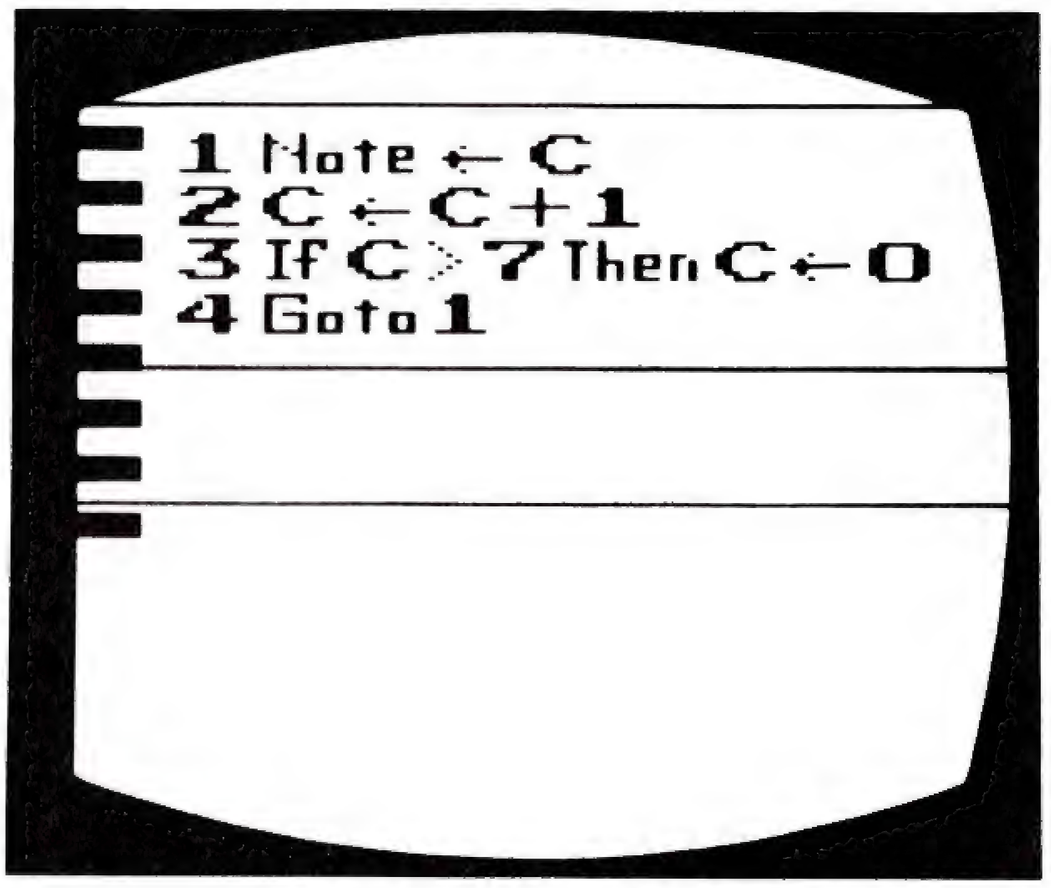
Stoppen Sie das Programm, und fügen Sie eine neue Zeile **2** ein (in der weißen Betriebsart, bringen Sie die Positionsanzeige zum Ende von Zeile **1** und drücken Sie **NEW LINE**. Die frühere Zeile **2** wird zur Zeile **3**).

```
2       If Note > 6 Then Note ←— 0
```

Ihre Anzeige sieht so aus:
(Wenn die **STACK** und **VARIABLES** Bereiche entfernt sind)

Mit den Befehlen **IF** und **THEN** weisen wir das Programm an, daß es, "wenn" (IF) etwas geschieht, es "dann" (THEN) etwas anderes machen muß. In diesem Falle, wenn die **IF** Note mehr als 6 ist, muß die **THEN** Note auf 0 geändert werden. Beginnen Sie mit dem Programm, und achten Sie auf die Abarbeitung im **STACK** Bereich. Wenn der Wert der Note die 7 erreicht, wird er höher als (>) 6, und das Programm ändert seinen Wert auf 0.

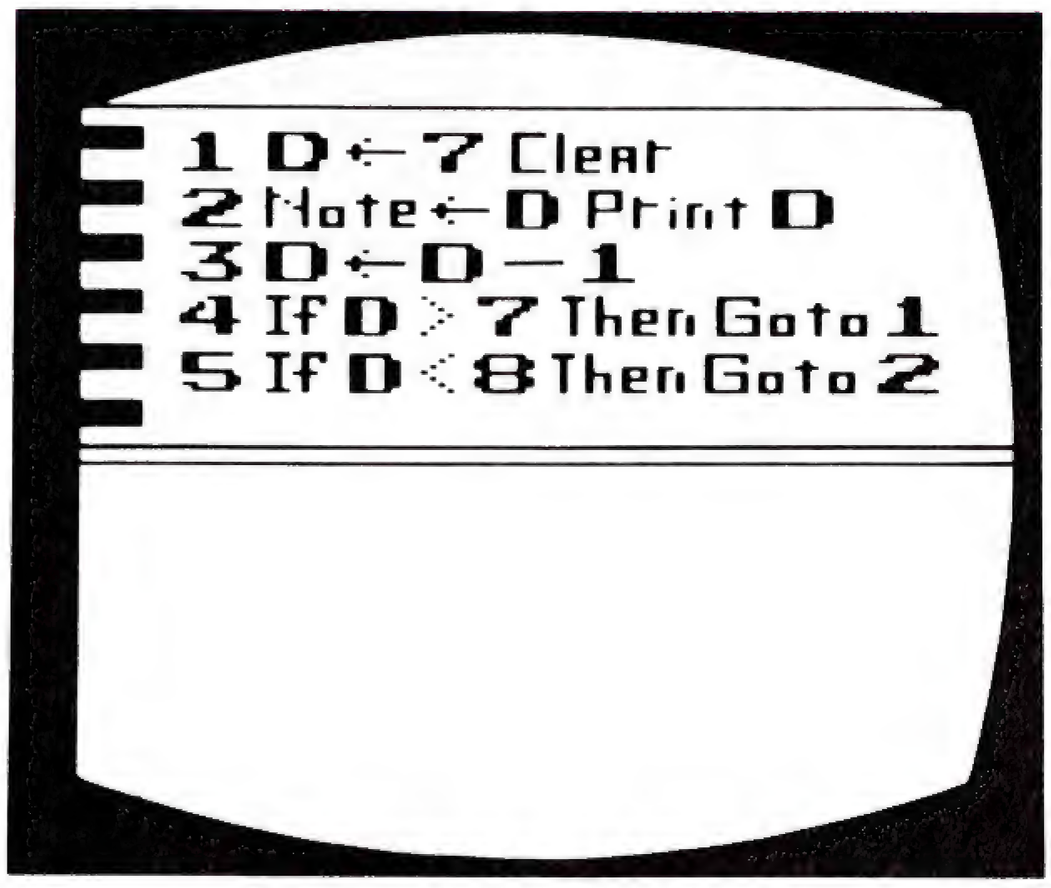
Wir wollen uns nun ein neues Programm vornehmen, um dieselben Resultate in anderer Form zu erzielen. Sie geben folgendes ein:



Fahren Sie das Programm. Sie werden sehen, daß Sie dieselben Resultate erhalten, außer, daß die Noten in gleichen Abständen erscheinen. Wenn Sie möchten, daß das Programm den Wert von **C** im **OUTPUT** Bereich anzeigt, geben Sie die Befehle **Print C** und **Clear** irgendwo im Programm ein. Um die Darstellung im **OUTPUT** Bereich, wenn das Programm den Wert zeigt, flimmerfrei zu machen, fügen Sie ein Komma (grüne Betriebsart) nach dem Wert in der Print Zeile, **Print C**, ein.

Für die Variablen des Programms können Sie jeden Buchstaben des Alphabets benützen. Das folgende

Programm hat sämtlichen Hauptfunktionen, die wir bis jetzt gelernt haben, und verwendet praktisch sämtlichen, im Computer verfügbaren Speicher. Nach Eingabe dieses Programmes entfernen Sie es von der Anzeige, und bringen den **STATUS, STACK** und **OUTPUT** Bereich auf den Bildschirm; dann fahren Sie das Programm.



In diesem Programm haben wir zwei **IF/THEN** Befehle erteilt. Sind die im ersten Befehl gesetzten Bedingungen nicht erfüllt, (Zeile 4), geht der Computer zur nächsten Zeile (Zeile 5). Ist genügend Speicher vorhanden, können mehrere Befehle zwischen jeweils zwei **IF/THEN** Befehlen eingegeben werden.

In manchen Fällen können zwei Befehle auf eine Zeile gebracht werden, wie in Zeile **1** und Zeile **2** im obigen Programm. Auf diese Weise können wir etwasSpeicher für später im Programm aufsparen.

ANWENDUNG DER FUNKTIONEN KEY UND PRINT

Die **KEY** Funktion (rote Betriebsart) wird benützt, um eine Variable einzugeben, während das Programm abläuft. Das Programm wertet **KEY** aus, und ersetzt es durch eine Zahl, die Sie von der rechten Tastaturseite aus eingeben. Wenn keine Zahl eingegeben wird, liest das Programm **KEY = 0**.

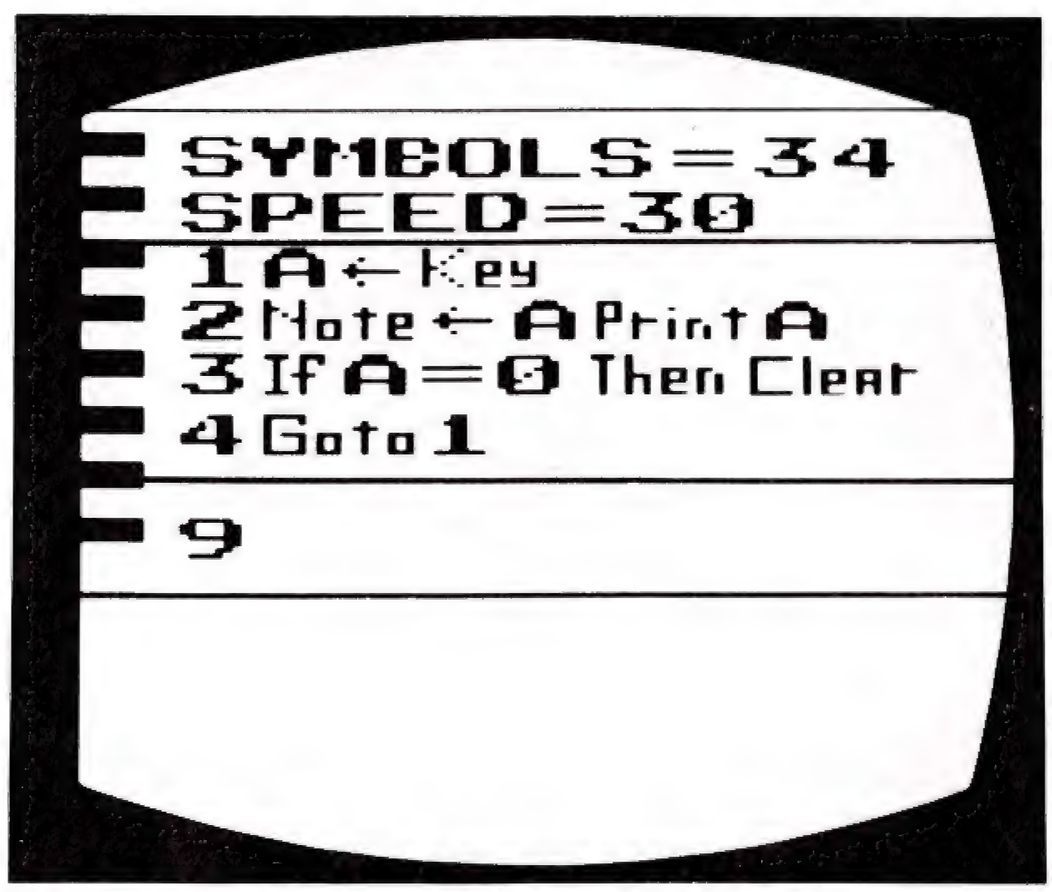
Hier ein einfaches Programm, das die **KEY** Funktion benützt:

Starten Sie das Programm, und drücken Sie einige der Tasten an der rechten Seite der Tastatur nieder.

Wenn Sie genug Übung haben, können Sie eine Melodie spielen.

Die **KEY** Funktion kann auch benützt werden, um Programmierung im **GRAPHICS** Bereich durchzuführen, wie wir später sehen werden.

Versuchen Sie dieses Programm mit Anwendung der **KEY**, **NOTE** und **PRINT** Funktionen:

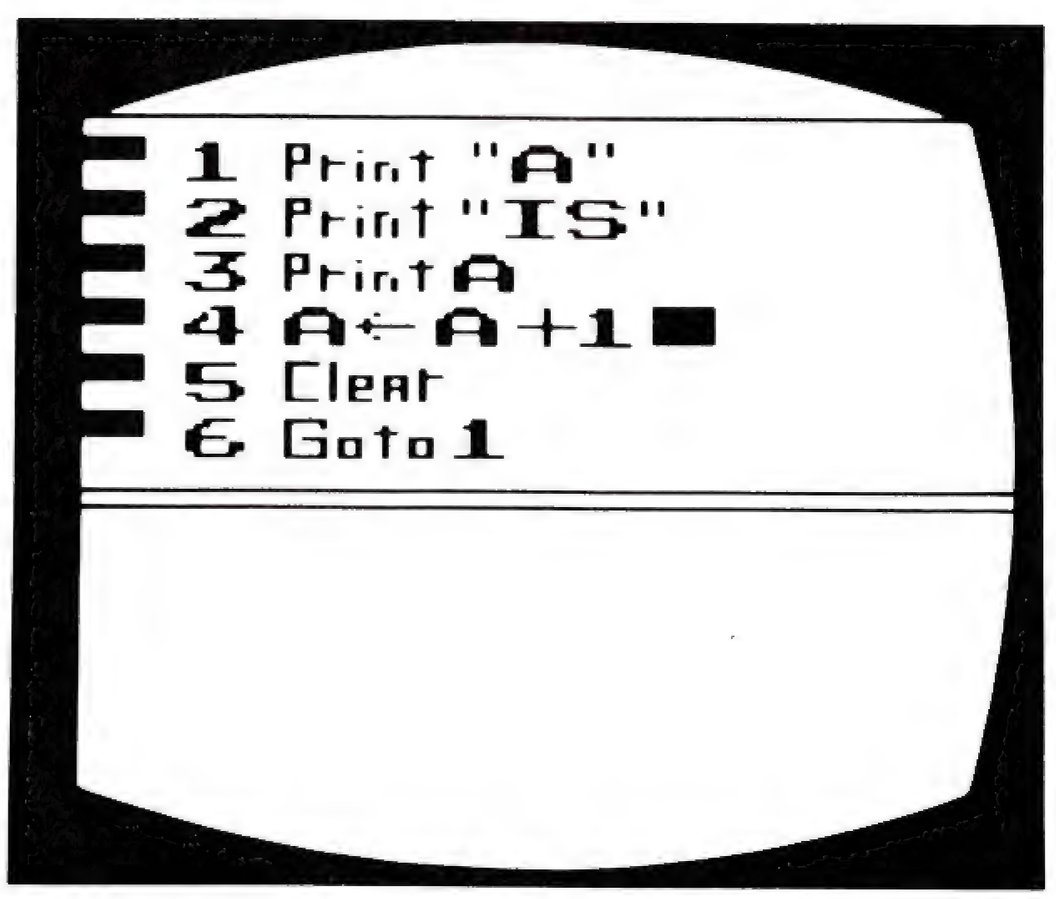


Beobachten Sie dieses Programm im **OUTPUT** Bereich. Wenn Sie eine Zahl von der rechten Seite der Tastatur aus eingeben, spielt das Programm diese Note, und stellt die Zahl im **OUTPUT** Bereich dar.

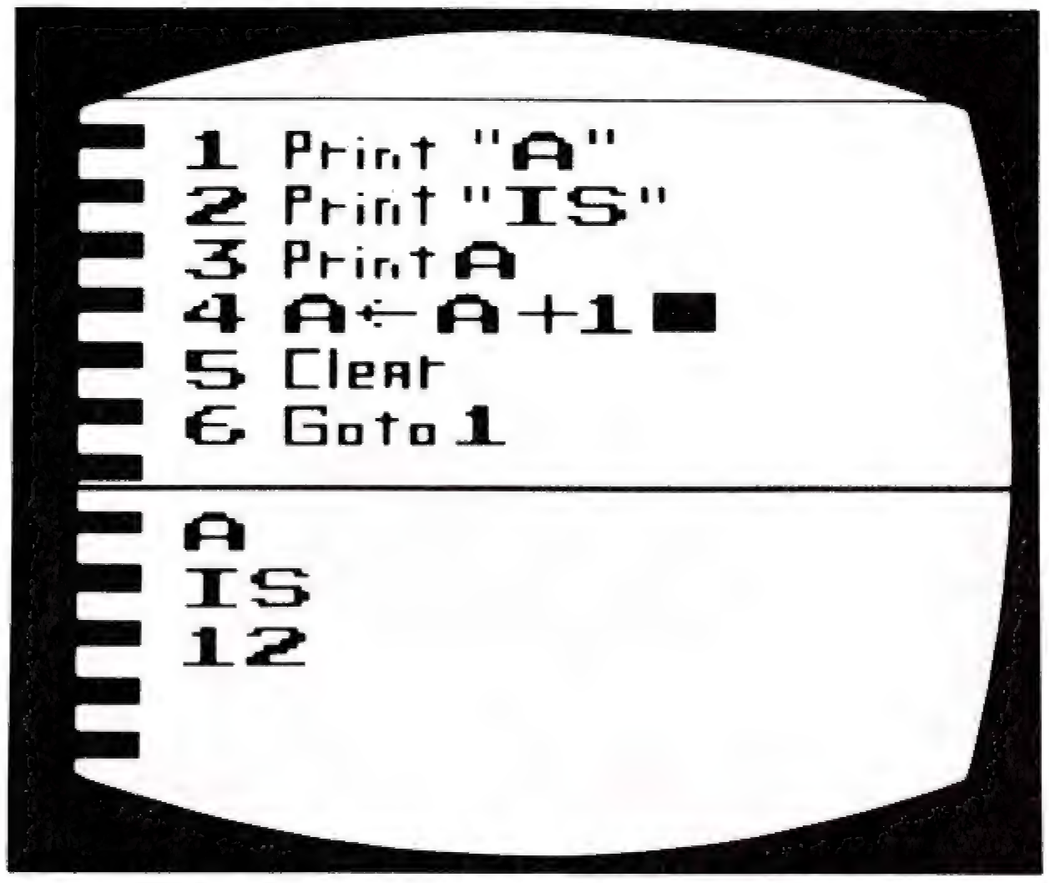
Und jetzt wollen wir das Programm ein bißchen ändern. Fügen Sie in Zeile 2 nach **Print A** ein Komma (,) ein, und starten Sie das Programm. Durch Einfügen des Kommas an dieser Stelle sagten Sie dem Programm, daß Sie auf dieser Zeile so viele Variablen wie möglich gedruckt haben möchten.

VERWENDUNG DER PRINT FUNKTION

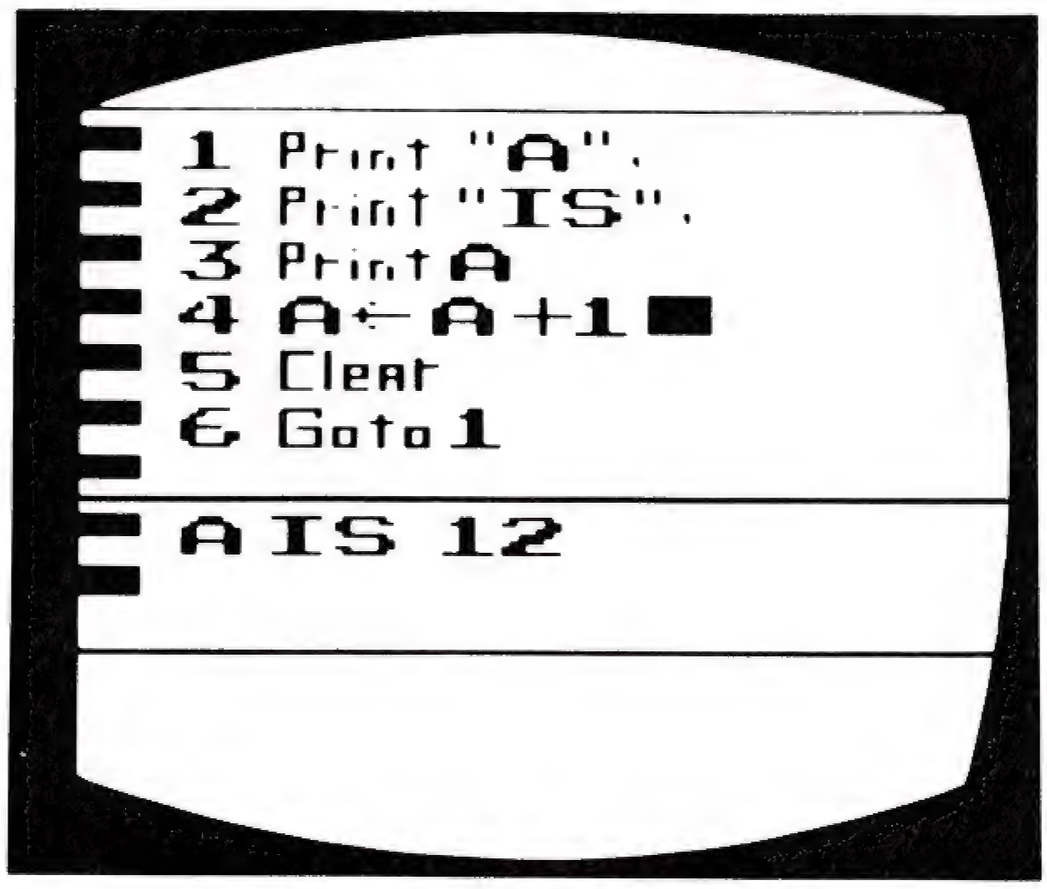
Wie in früheren Programmen aufgezeigt, können Sie die **PRINT** Funktion dazu benützen, dem Programm den Befehl zu erteilen, den Wert einer Variablen im **OUTPUT** Bereich zu drucken. Die **PRINT** Funktion kann auch benützt werden, um dem Programm den Befehl zu erteilen, Worte auf der Anzeige zu drucken. Anzuzeigende Worte müssen in Anführungszeichen (") erscheinen. Setzen Sie die Geschwindigkeit (**SPEED**) auf **8** und geben Sie folgendes Programm ein:



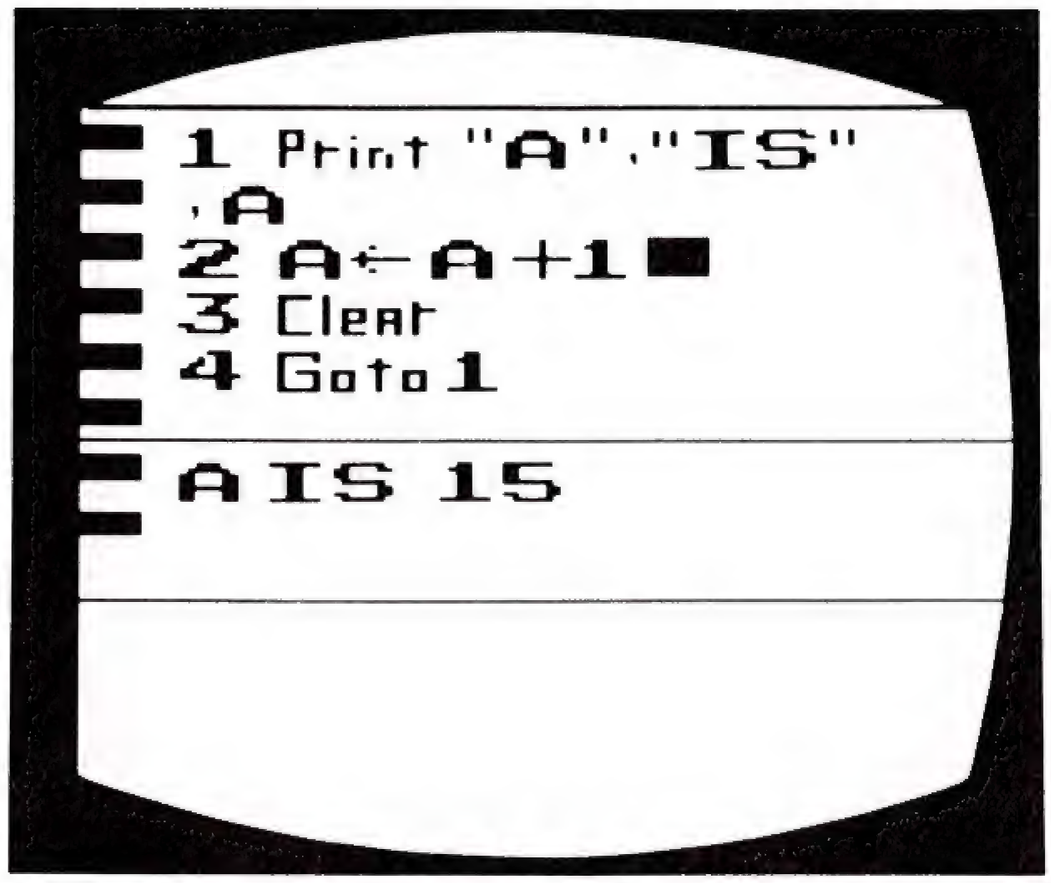
Jetzt fahren Sie das Programm, und beobachten den **OUTPUT** Bereich. Das Programm druckt die von Ihnen eingegebenen Worte, jedoch "stacked".



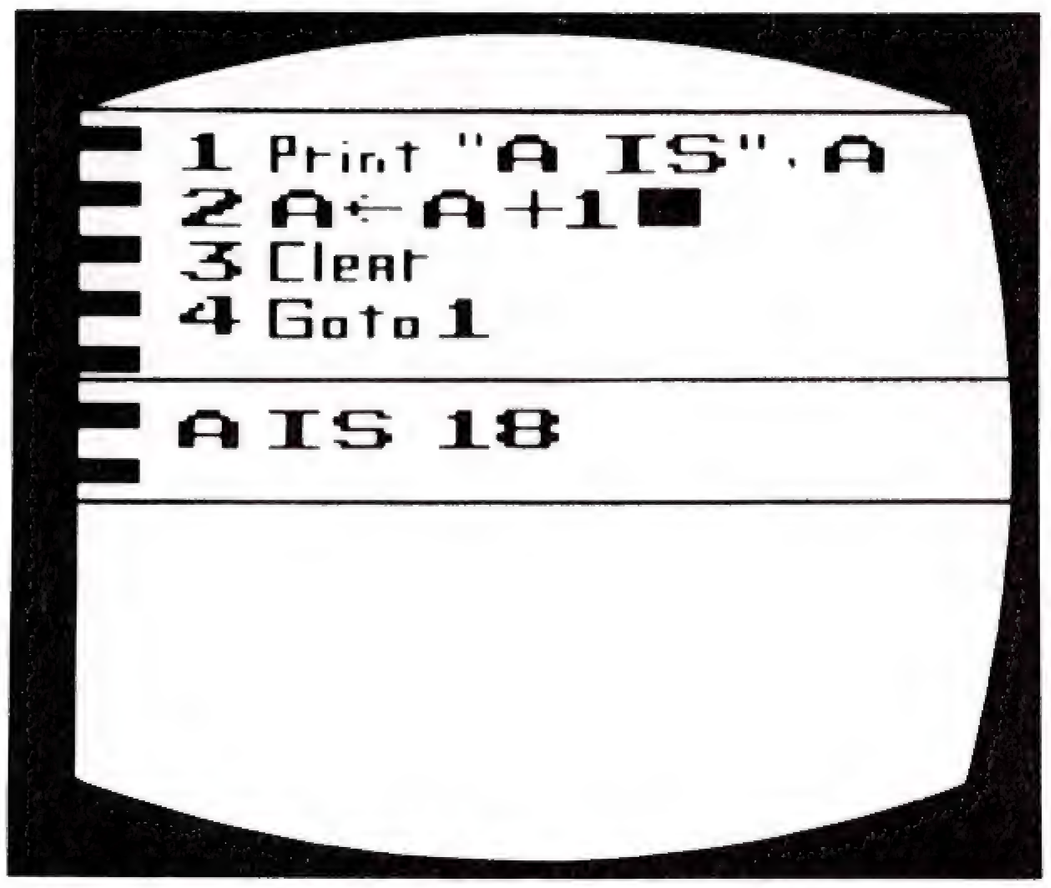
Wir ändern das Programm nun geringfügig, indem wir ein Komma (,) am Ende von Zeile **1** und Zeile **2** eingeben. Das Programm sieht nun folgendermaßen aus:



Das Komma (,) in Zeile **1** hat das Programm instruiert, daß alles, was auf Zeile **1** ist, im **OUTPUT** Bereich auf derselben Zeile wie der Befehl auf Zeile **1** dargestellt werden soll. Das Komma auf Zeile **2** hat das Programm instruiert, daß der Befehl der Zeile **3** der Zeile **2** folgen soll. Diese Befehle können gekürzt werden, um das Programm zu ändern:



Das Programm kann weiter gekürzt werden:



Das Schreiben des Programms in dieser Form spart außerdem etwas Speicher ein.

ANWENDUNG DES GRAPHICS BEREICHES

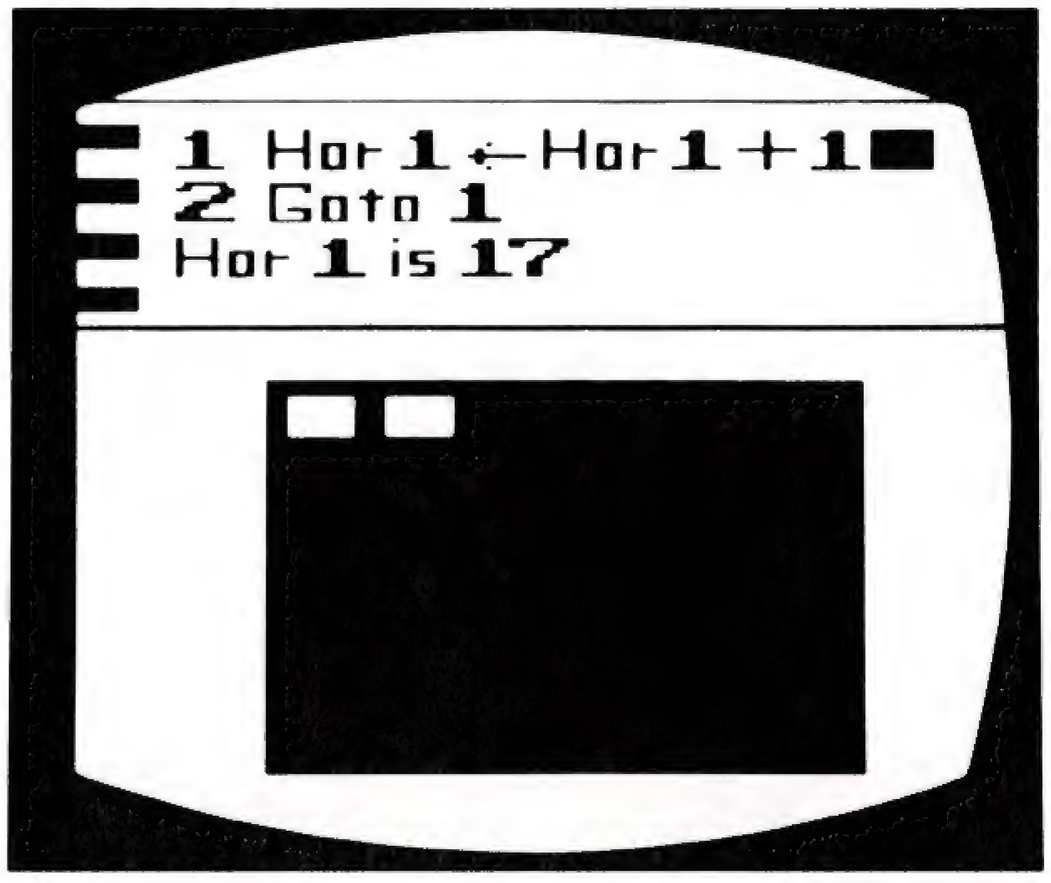
Der **GRAPHICS** Bereich ist das blaue, rechteckige Feld auf dem Fernsehbildschirm. Auf den ersten Blick erscheint es, als enthalte es ein rotes Quadrat in der oberen linken Ecke. In der Tat verdeckt das rote Quadrat ein weißes Quadrat, und beide können im Rahmen Ihrer Programmsteuerung unabhängig voneinander am Feld bewegt werden.

Um die Quadrate zu bewegen, müssen Sie Ihre Koordinaten ändern. Das rote Quadrat ist Objekt Nummer 1. Seine horizontale Koordinate wird an der Tastatur durch **Hor 1** und seine vertikale Koordinate durch **Ver 1** dargestellt. **Hor 2** und **Ver 2** sind die Koordinaten von Objekt Nummer 2, dem weißen Quadrat.

Beide Objekte, oder Quadrate, beginnen in der oberen linken Ecke des blauen Feldes, wobei nur das rote Quadrat sichtbar ist. Die obere linke Ecke ist die Null (0) Position oder Anfangsstellung. Wenn die Variablen **Hor 1**, **Ver 1**, **Hor 2** und **Ver 2** nicht definiert sind (ohne gegebenen Wert), haben sie den Wert = 0 (weshalb die Quadrate in der Stellung in der oberen linken Ecke bleiben).

Wird einer Koordinate ein anderer Wert als 0 zugewiesen, (z.B. **Hor 1 ← 10**), springt das entsprechende Objekt zur entsprechenden Stellung am blauen Feld, wenn das Programm abläuft.

Geben Sie folgendes Programm ein:



Dieses Programm bewegt das rote Quadrat nach rechts, jeweils um eine horizontale Stelle. Fahren Sie das Programm, und Sie werden erkennen, daß sich das rote Quadrat langsam nach rechts bewegt. Beobachten Sie den **VARIABLES** Bereich und Sie sehen, daß der Wert von **Hor 1** größer wird. Er wächst, bis er **99** — die höchstmögliche Zahl — erreicht, und dann auf **0** zurückfällt. Wenn **Hor 1** auf **99** angestiegen ist, wird das rote Quadrat ganz rechts am blauen Feld angelangt sein. An dieser Stelle verschwindet das rote Quadrat und taucht ganz links am Feld mit **Hor 1 = 0** wieder auf.

Die horizontalen Koordinaten (**Hor 1** und **Hor 2**) liegen also im Bereich zwischen 0 und 99. Die vertikalen Koordinaten (**Ver 1** und **Ver 2**) liegen gleichermaßen im Bereich zwischen 0 und 99, wobei 0 die Stellung oben am blauen Feld, und 99 die Stellung unten ist.

Geben Sie folgendes Programm ein:

```
1 Hor 1 ← Hor 1 + 6
2 Ver 1 ← Ver 1 + 8
3 Hor 2 ← Hor 2 + 4
4 Ver 2 ← Ver 2 + 4
5 If Hit Then Note ← 2
Else Note ← 7
6 Goto 1
```

Nehmen Sie das Programm vom Bildschirm, und überzeugen Sie sich, daß der **GRAPHICS** Bereich ganz sichtbar ist. Stellen Sie die Geschwindigkeit (**SPEED**) = **60**, und fahren Sie das Programm. Dieses Programm zeigt Ihnen ein Verfahren, die Quadrate am Feld zu bewegen. Es zeigt außerdem, wie die **HIT** und **ELSE** Funktionen angewendet werden können.

In Zeile **5** instruiert das Programm den Computer, die Note **2** erklingen zu lassen **IF** (WENN) die Quadrate **HIT**, (ZUSAMMENSTOSSEN) was periodisch eintritt. Zum **HIT** müssen die Quadrate dieselben Koordinaten belegen. **ELSE** (Ansonsten) wird der Computer angewiesen, die Note **7** zu spielen; dies führt der Computer solange aus, bis die Quadrate anstoßen (**HIT**).

(Entfernen Sie **ELSE NOTE ← 7** von der Zeile **5** wird die Note **2** gespielt, wenn die Quadrate "hit" (zusammenstoßen) und keine weitere Note angeschlagen wird.

ANWENDUNG DER MOD FUNKTION

Mod ist ein mathematisches Operationssymbol, ähnlich dem Divisionszeichen (÷). Bei **BASIC PROGRAMMING** wird Ganzzahl-Division benutzt, d.h. es wird nur mit ganzen Zahlen gearbeitet, ohne Divisionsreste. Wieviel ist zum Beispiel 14 ÷ 5? Sie könnten es ausdrücken als 2 und 4/5, bleibt 4. Im **BASIC PROGRAMMING** ergibt 14 ÷ 5 = 2. Obwohl 14 durch 5 zweimal geteilt wird (woraus sich 10 ergibt), mit einem Rest von 4, benützt **BASIC** nur ganze Zahlen, daher ist die Antwort, die Sie erhalten, 2.

Der Computer gibt dieselbe Antwort für 12 ÷ 4 wie er es für 13 ÷ 4 gibt. In beiden Fällen ist die Antwort 3, da 12 geteilt durch 4 in beiden Fällen 3 ganze Male ergibt. Der Computer erkennt den Rest von 1 im Falle von 13 ÷ 4 nicht an.

Und jetzt zur **Mod** Funktion. **Mod** erhält den Rest, der nach der Division der ersten Zahl in die zweite übrig bleibt. So ist **14 Mod 5 = 4**. 5 geht in 14 zweimal (ergibt 10) und 4 bleibt übrig. **Mod** erhält den Rest, und daher ist **Mod 4**.

Was ist **13 Mod 4**? Die Antwort ist **1**, da **1** übrig bleibt, nachdem **13** durch **4** dividiert wird (was **12** ergibt). Wie ist es mit **12 Mod 4**? In diesem Fall beträgt **Mod = 0**, da 12 dividiert durch 4 keinen Rest ergibt, also 0 übrig bleibt.

Normalerweise ist eine Division durch 0 mathematisch gesprochen nicht definiert. In **BASIC PROGRAMMING** ergibt eine Division durch 0 also das Resultat 0, so daß $5 \div 0 = 0$ ist.

OPERATIONSSYMBOL-PRIORITÄTEN

In einer Gleichung werden die arithmetischen Operationssymbole, (+, −, ×, ÷, Mod, usw.) in der Reihenfolge einer festgelegten Priorität gerechnet. **BASIC PROGRAMMING** verwendet folgende Operationssymbol-Richtlinien:

1. × ÷ (höchste)
2. + −
3. Mod
4. =
5. ⟵ (niedrigste)

VERWENDUNG VON KLAMMERN

Die Verwendung von Klammern (grüne Betriebsart, linke Seite der Tastatur) gibt jeder in Klammern stehenden Zahl Priorität, wenn eine Gleichung ausgerechnet wird.

Wird zum Beispiel diese Gleichung ausgerechnet:

A 5 + 3 × 2 Mod 7

ist der erste Schritt: $3 \times 2 = 6$
der zweite Schritt: $5 + 6 = 11$
der dritte Schritt: 11 Mod 7 = 4
A = 4

Dieselbe Gleichung jedoch wird mit Klammern anders ausgerechnet, und ergibt eine andere Antwort für A:

A (5 + 3) × Mod 7

Erster Schritt: $5 + 3 = 8$
Zweiter Schritt: $8 \times 2 = 16$
Dritter Schritt: 16 Mod 7 = 2
A = 2

MUSTER-PROGRAMME

Hier finden Sie einige Muster-Programme. Sie werden sehen, wie viele Möglichkeiten Ihnen zur Verfügung stehen.
Vergessen Sie nicht, genau darauf zu achten, wie sich ein individueller Befehl (**IF, THEN, HIT, ELSE, PRINT, KEY** usw.) auf ein Programm auswirkt. Denken Sie auch daran, daß ein **KEY** Befehl, um das Programm erfolgreich ausführen zu können, von Ihrer Eingabe von der rechten Seite der Tastatur abhängen kann.

Jedesmal, wenn ein Programm verwirrend erscheint, stoppen Sie es, stellen es zum Anfang zurück und

gehen es Schritt für Schritt durch. Beobachten Sie das Programm im **STACK** Bereich, und sehen Sie, wie der Computer jeden einzelnen Schritt durchführt, sollte es Ihnen nicht schwerfallen, den Vorgang und seinen Ablauf zu verstehen.

Nachdem Sie die Programme gefahren und sich mit den Grundsätzen der Computer-Programmierung vertraut gemacht haben, sind Sie bald soweit, eigene Programme zu schreiben.

HIT

```
1 Clear PrintHit
2 Hor 2 ← Hor 2 + 1 Mod
3 3
3 Goto 1
```

MUSIC

```
1 Note ← If Note > 4
Then Note — 11 Else
Note + 1
2 Note ← Note + 2
3 Note ← Note — 2
4 Goto 1
```

```
1 A ← Key
2 If A > 0 Then Note ← A
— 1
3 Goto 1■
```

```
1 Note ← Note + Key +
7
2 Goto 1■
```

```
1 A ← A + 1
2 Note ← A
3 If (A Mod 2) = 0 Then
Note ← 0 Else Note ← 4
4 Goto 1■
```

CLOCK-LIKE PROGRAM

Wenn Sie das Clock Programm fahren, bringen Sie nur den **OUTPUT** Bereich auf den Bildschirm. Die Geschwindigkeit (**SPEED**) kann auf 30 oder 60 eingestellt werden.

```
1 Clear
2 A←A+1 Mod 60
3 If A=0 Then B←B+1
Mod 60
4 Print B.A
5 Goto 1■
```

PONG® SPIEL
(ohne Ton)

```
1 Hor 2←Hor 2+Key
2 If Ver 1>90 Then
Ver 1←88
3 If Hit Then Ver 1←3
4 Ver 1←Ver 1+If
Ver 1 Mod 2 Then 8 Else
32
5 Hor 1←Hor 1+7
6 Goto 1■
```

PONG® SPIEL
(Ball und Schläger)

```
1 Hor 1←Hor 1+Key
2 Hor 2←Hor 2+8
3 Ver 2←Ver 2-3
4 If Hit Then Ver 2←93
 .Note←7
5 Goto 1
```

1 VIDEO GIOCO

BASIC PROGRAMMING

INTRODUZIONE

Il **BASIC PROGRAMMING** è stato concepito per insegnare le fasi essenziali della programmazione degli elaboratori elettronici. **BASIC** è la sigla composta dalle iniziali di *Beginners All-purpose Symbolic Instruction Code* (Codice di istruzioni simbolico multiuso per principianti), un linguaggio messo a punto per consentire un facile apprendimento della "scrittura" dei programmi per elaboratori.

I suddetti programmi consistono essenzialmente in una serie di istruzioni. Essi governano il flusso delle informazioni attraverso il computer. Il **BASIC PROGRAMMING** consente di immettere nel Video Computer System™ le istruzioni di cui esso ha bisogno per svolgere delle semplici funzioni.

Si tenga presente che il **BASIC PROGRAMMING** è dotato di una memoria limitata in confronto ai sistemi di informatica più sofisticati, ma costituisce pur sempre un ottimo strumento per l'apprendimento delle nozioni fondamentali della programmazione degli elaboratori.

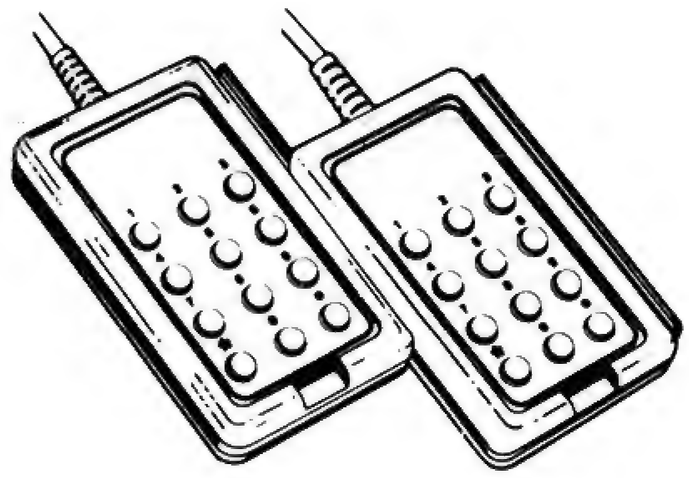
NOTA BENE: Portare sempre l'interruttore di alimentazione **power** nella posizione **off** quando si inserisce o si rimuove una cartuccia Game Program™ ATARI. In tal modo si proteggono i componenti elettronici e si prolunga la vita utile del Video Computer System ATARI.

Questo Game Program™ può provocare un lieve "rullio" dell'immagine sullo schermo di alcuni televisori. In tal caso può essere necessario intervenire sull'apposito comando di regolazione del sincronismo verticale.

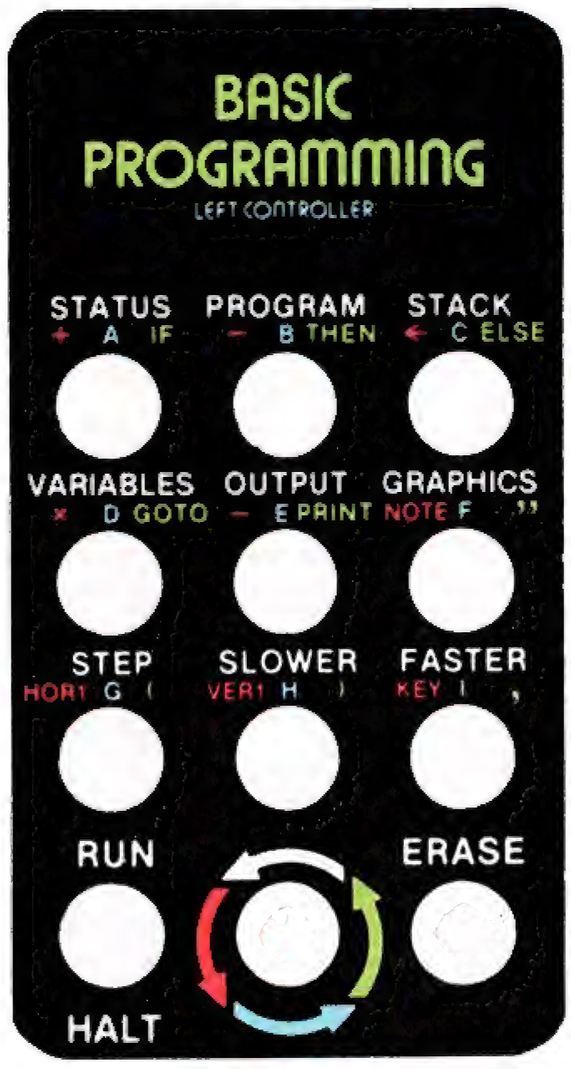
COMANDI A TASTI

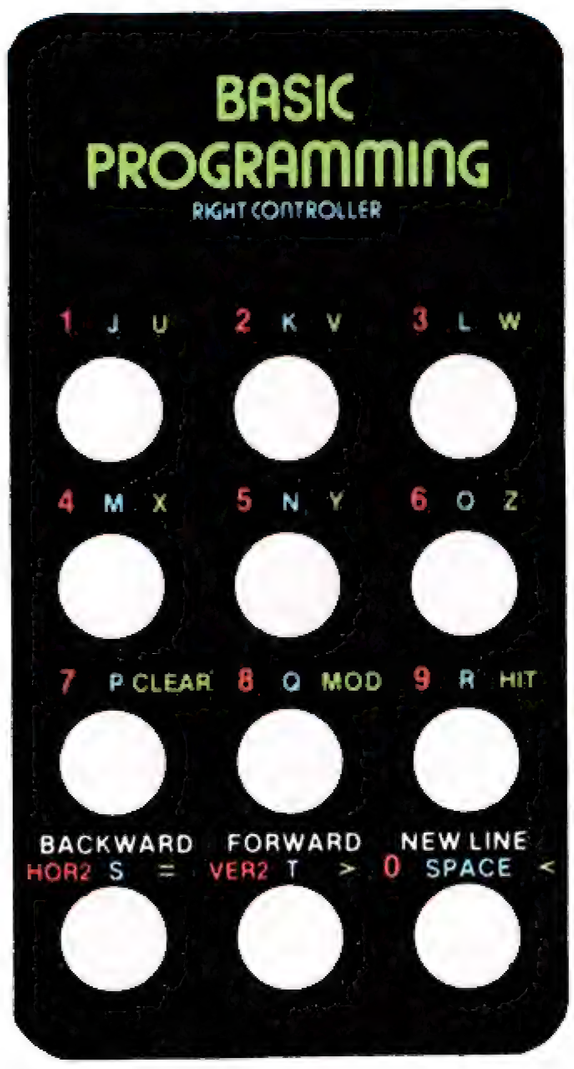
Con questa cartuccia Game Program™ ATARI si usano i comandi a tasti. Assicurarsi che i relativi cavi siano stabilmente inseriti nelle prese **CONTROLLER** sul retro del Video Computer System ATARI.

Per maggiori particolari, vedi Parte 3 del Manual di istruzioni del Video Computer System.



Per collegare i due comandi, inserire la linguetta del comando di sinistra nella guida scanalata del comando di destra, come mostrato in figura. Così uniti, i due comandi formano una tastiera a 24





tasti utilizzabile per immettere i programmi nel computer e per visualizzare sullo schermo le presentazioni che interessano.

Estrarre le targhette dei comandi dalla busta. Applicare la targhetta con la scritta **LEFT** sul comando inserito nella presa **LEFT CONTROLLER** sul retro della console del computer, e quella con la scritta **RIGHT** sul comando inserito nella presa **RIGHT CONTROLLER**.

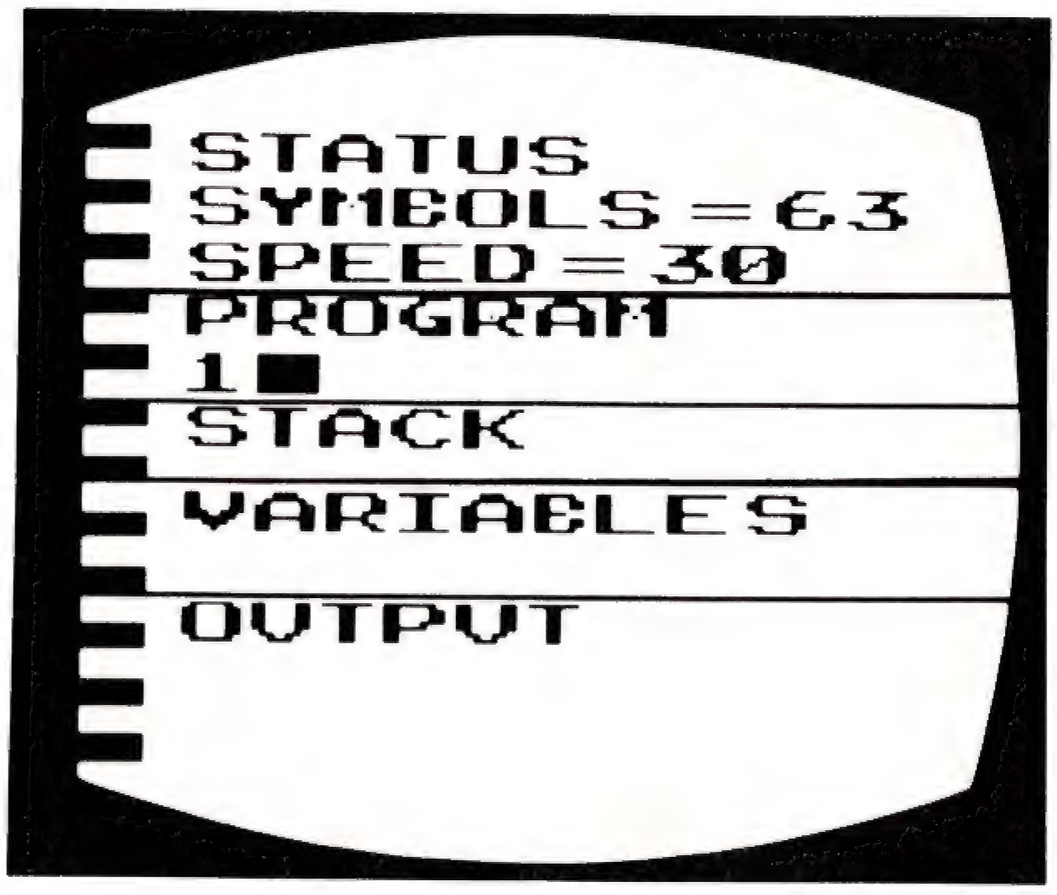
ZONE DEL VIDEO

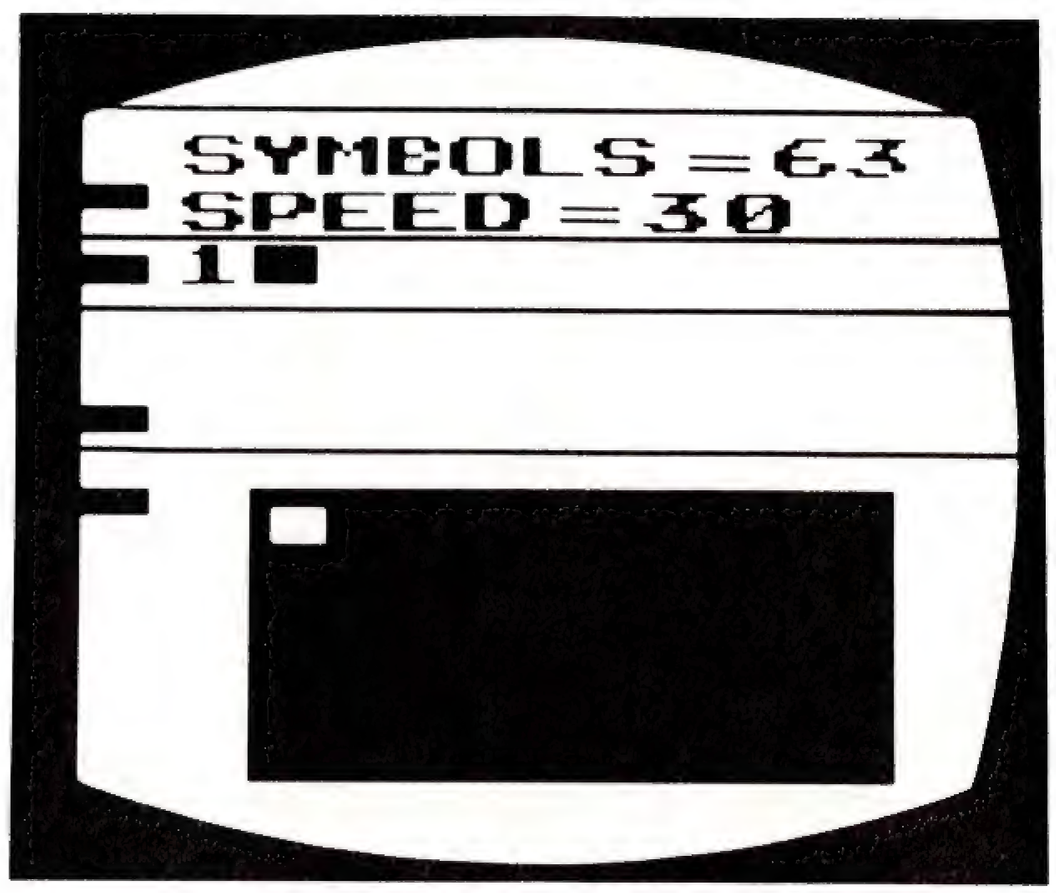
Ai fini della visualizzazione, il video è suddiviso in sei regioni:

1. La zona **PROGRAM** (programma), utilizzata per dare le istruzioni al computer.
2. La zona **STACK** (catasta), in cui vengono presentati i risultati temporanei del programma mentre il computer lo esegue.
3. La zona **VARIABLES** (variabili), in cui vengono visualizzati i valori di ciascun dato variabile del programma mentre il computer lo esegue.
4. La zona **OUTPUT** (uscita) in cui vengono presentati i dati di uscita mentre si esegue il programma.
5. La zona **STATUS** (stato), in cui viene indicata la memoria disponibile in qualsiasi momento per l'elaborazione del programma, come anche la velocità alla quale quest'ultimo viene eseguito.
6. La zona **GRAPHICS** (grafica) è dotata di due rettangolini colorati che possono essere spostati in vari punti a seconda del programma.

Prima di cominciare il programma, portare il commutatore **left difficulty** nella posizion **b**. Sullo schermo appariranno le varie zone del video. Quando il suddetto commutatore è nella posizione **a**, dallo schermo compaiono i nomi di ciascuna zona e viene mostrata la zona GRAPHICS.

Il commutatore **right difficulty** non ha alcuna funzione in questo Game Program.





IL CURSORE

Portare il commutatore **left difficulty** su **b** e quindi portare l'interruttore **power** prima su **off** e poi su **on**. Nella regione PROGRAM si trova un rettangolo bianco: il cursore. Premere quattro volte il tasto **shift control** al centro dell'ultima fila di tasti del comando **LEFT CONTROLLER**: il colore del cursore cambia, nell'ordine, da bianco a rosso a blu a grigio, tornando infine a bianco.

Il cursore serve per immettere il programma nel computer. Ciascuno dei suoi quattro colori, impostati attraverso il tasto **shift control**, corrisponde a quello dei tasti d'ingresso (inputs) del comando. Il bianco serve per impartire comandi al computer, gli altri colori per introdurre simboli nel programma.

COME SPOSTARE IL CURSORE DA UNA ZONA ALL'ALTRA DEL VIDEO

Assicurarsi che il commutatore **left difficulty** sia nella posizione **b** e portare l'interruttore **power** prima su **off** e poi ancora su **on**. Premere il tasto **FORWARD** e il cursore si sposterà dalla zona **PROGRAM** a quella **STACK**; premendo una seconda volta il tasto **FORWARD**, il cursore passa da **STACK** a **VARIABLES**; premendolo una terza volta, il cursore si sposta da **VARIABLES** ad **OUTPUT**.

Premendo il tasto **BACKWARD** si riporta il cursore nella zona **PROGRAM**. I tasti **FORWARD** e **BACKWARD** possono essere premuti una sola volta per ciascuna zona oppure possono essere tenuti abbassati.

Portare ora il commutatore **left difficulty** nella posizione **a**. Scompariranno i nomi delle singole zone e comparirà una parte della zona **GRAPHICS**. Portare nuovamente il cursore in ciascuna delle altre zone; si noterà che esso non passa più nella zona **GRAPHICS.**

COME ELIMINARE LE SINGOLE REGIONI DAL VIDEO

Portare il commutatore **left difficulty** nella posizione **b** e portare l'interruttore **power** della console prima su **off** e quindi su **on**. Sul lato sinistro del comando a tasti si trovano una serie di tasti (colore bianco) corrispondenti rispettivamente alle seguenti zone: **STATUS, PROGRAM, STACK, VARIABLES, OUTPUT** e **GRAHPICS**. Cominciando dalla zona **STATUS**, premere il tasto **STATUS** una sola volta per fare scomparire la zona stessa dal video e far passare la zona **PROGRAM** in cima al video.

Premendo il tasto **PROGRAM**, la zona **PROGRAM** scompare e la zona **STACK** si sposta in cima al video. Premendo il tasto **STACK**, la zona omonima scompare e la zona **VARIABLES** si sposta in cima al video. Premendo il tasto **VARIABLES**, la corrispondente zona scompare e la zona **OUTPUT** si porta in cima al video.

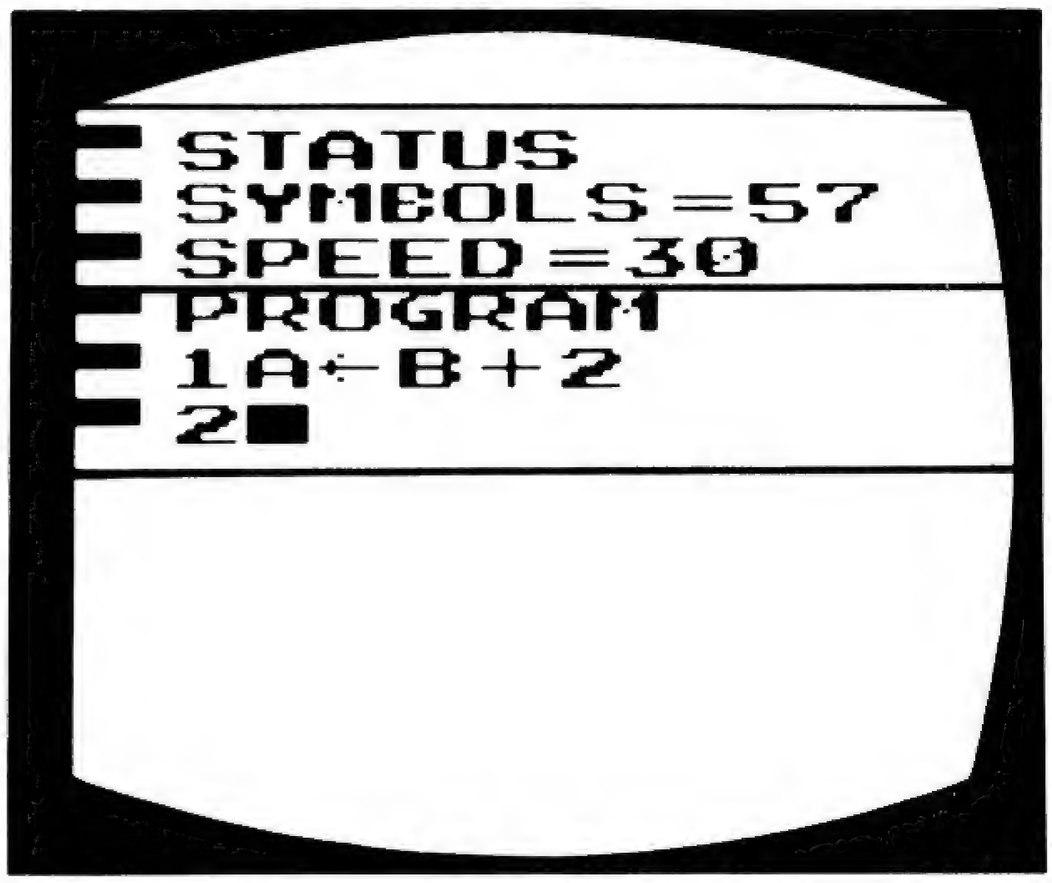
Per eliminare la zona **OUTPUT** si agisce sul tasto omonimo, facendo apparire per intero la zona **GRAPHICS**. Premendo il tasto **GRAPHICS**, la corrispondente zona scompare. A questo punto sul video non vi sono più zone.

Premere ora il tasto **STACK**, in modo da far riapparire sul video la zona omonima. Eliminare tale zona e spostare verso l'alto le zone **OUTPUT** e **DISPLAY**. Con il commutatore **left difficulty** nella posizione **a** o **b** si può visualizzare od eliminare qualsiasi zona.

È importante notare che ai fini dell'esecuzione del programma è irrilevante che vengano visualizzate o meno le varie zone.

ESECUZIONE DEL PROGRAMMA

Proviamo un semplice programma. Assicuriamoci che il commutatore **left difficulty** sia nella posizione **b** e portiamo l'interruttore **power** prima **off** e poi **on**. (Un'altro modo di cancellare un programma ed azzerare tutti i valori consiste nello spingere in basso il commutatore **game select**. Azionando il commutatore **game reset** si cancellando tutti i valori e si ripristina il programma dall'inizio senza cancellarlo.)



Eliminiamo dallo schermo le zone **STACK, VARIABLES, OUTPUT** e **GRAPHICS**. Il cursore si troverà nel modo bianco ed alla destra del numero **1** nella zona **PROGRAM**.

Portiamo il cursore nel modo blu e premiamo il tasto **A**. Sul video apparirà la lettera **A** a fianco del numero **1**. (Ciascuna linea è numerata, per consentire di vedere dove finisce l'una e comincia l'altra.) Portiamo ora il cursore nel modo rosso e premiamo il tasto ← . A fianco della lettera **A** apparirà una freccetta. Ritorniamo quindi al modo blu ed impostiamo **B**. Portando il cursore nel rosso, impostiamo + ed il numero **2**. Torniamo ora al bianco ed immettiamo **New Line** (nuova linea). Per cominciare una nuova linea è in ogni caso necessario trovarsi nel modo bianco.

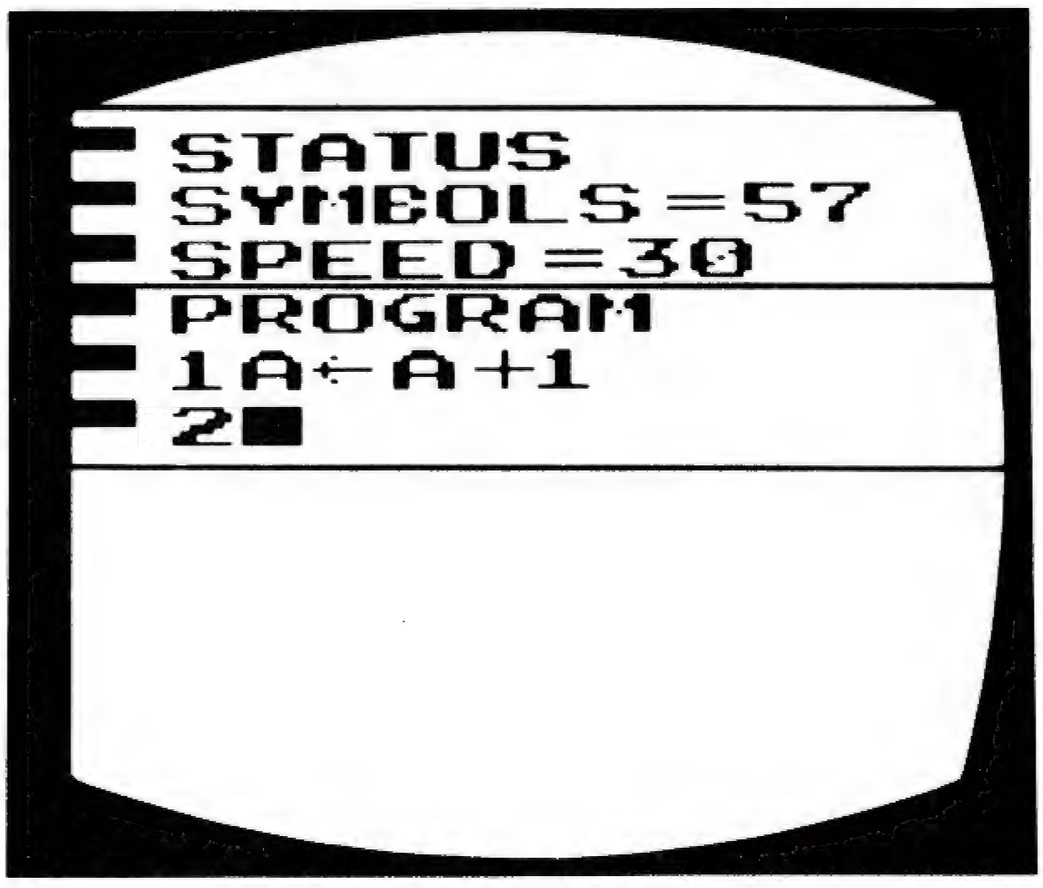
Il video dovrà risultare quindi come mostrato in figura.

Se si commette un errore durante la programmazione, lo si può cancellare mediante il tasto **ERASE**. Si noti che questo tasto **non** è a codice di colore e può pertanto essere utilizzato quando il tasto **shift** è su un modo di un altro colore.

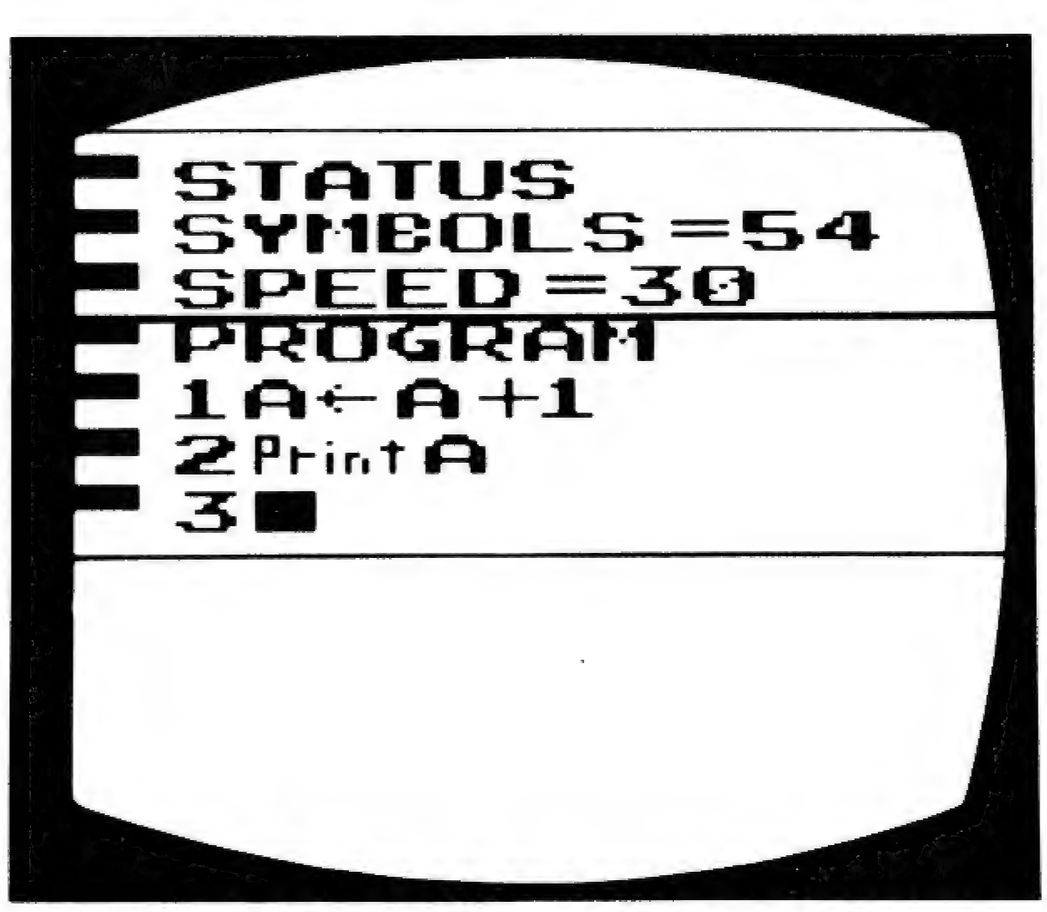
Utilizzando la linea precedentemente impostata nel programma, procediamo con un esercizio. Premiamo il tasto **ERASE** una sola volta: il cursore si porterà direttamente dalla linea **2** alla destra del numero **2** nella linea **1**. Premiamo una seconda volta il tasto **ERASE** per cancellare il **2** dal programma. Impostiamo ora il numero **1** utilizzando il modo rosso. Portiamo quindi il cursore nel modo bianco e spostiamolo, mediante il tasto **BACKWARD**, fino a quando non risulti immediatamente alla destra della lettera **B** del programma. Per eliminare tale lettera, premiamo il tasto **ERASE** e sostituiamola quindi con la lettera **A** (modo blu).

Nel modo bianco, azionare il tasto **FORWARD** per spostarsi al termine della linea **1** ed impostare **New Line**.

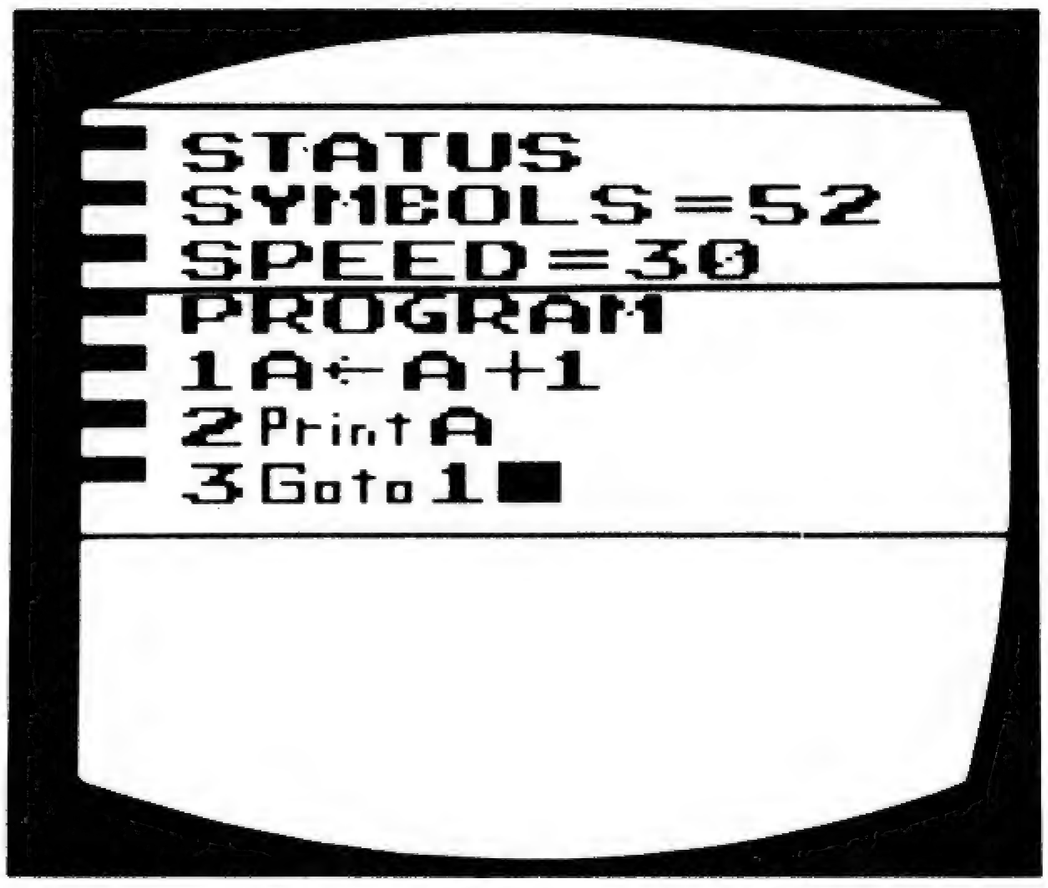
Ricordiamoci che il cursore non deve trovarsi sul simbolo da cancellare ma immediatamente alla destra di questo. Il video dovrà presentarsi come mostrato in figura.



Il cursore si trova alla destra del numero **2** nella linea **2**.
Impostiamo ora **PRINT** nel modo verde. Successivamente, nel modo blu, impostiamo **A** e passiamo a una nuova linea. Il video risulterà come mostrato in figura.

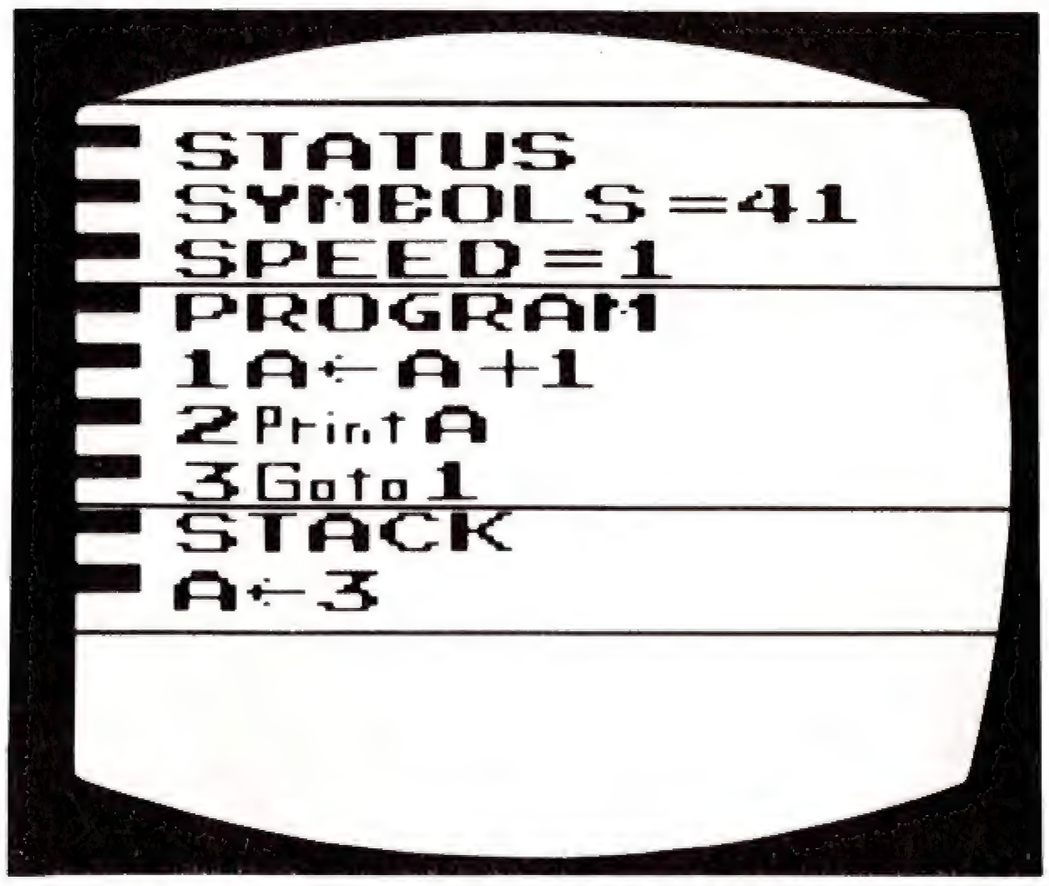


Con il cursore nel modo verde, premiamo il tasto **GO TO** e quindi impostiamo **1** nel modo rosso. Prima di tutto, controlliamo ancora una volta il video.



Si nota che nella zona **STATUS** ci rimangono 52 "bytes" o simboli di memoria. La velocità indicata è di **30 (SPEED = 30)**. Portiamo il cursore nel modo bianco e premiamo il tasto **SLOWER**. Si noti che ogni volta che azioniamo questo tasto la velocità diminuisce. Premendo il tasto **FASTER**, la velocità aumenta.

Prima di cominciare il programma, impostiamo **SPEED = 1**. Premiamo quindi il tasto **STACK**. Spingiamo due volte il tasto **RUN/HALT** ed il programma comincia. È così possibile osservare l'esecuzione di ciascuna parte del programa.



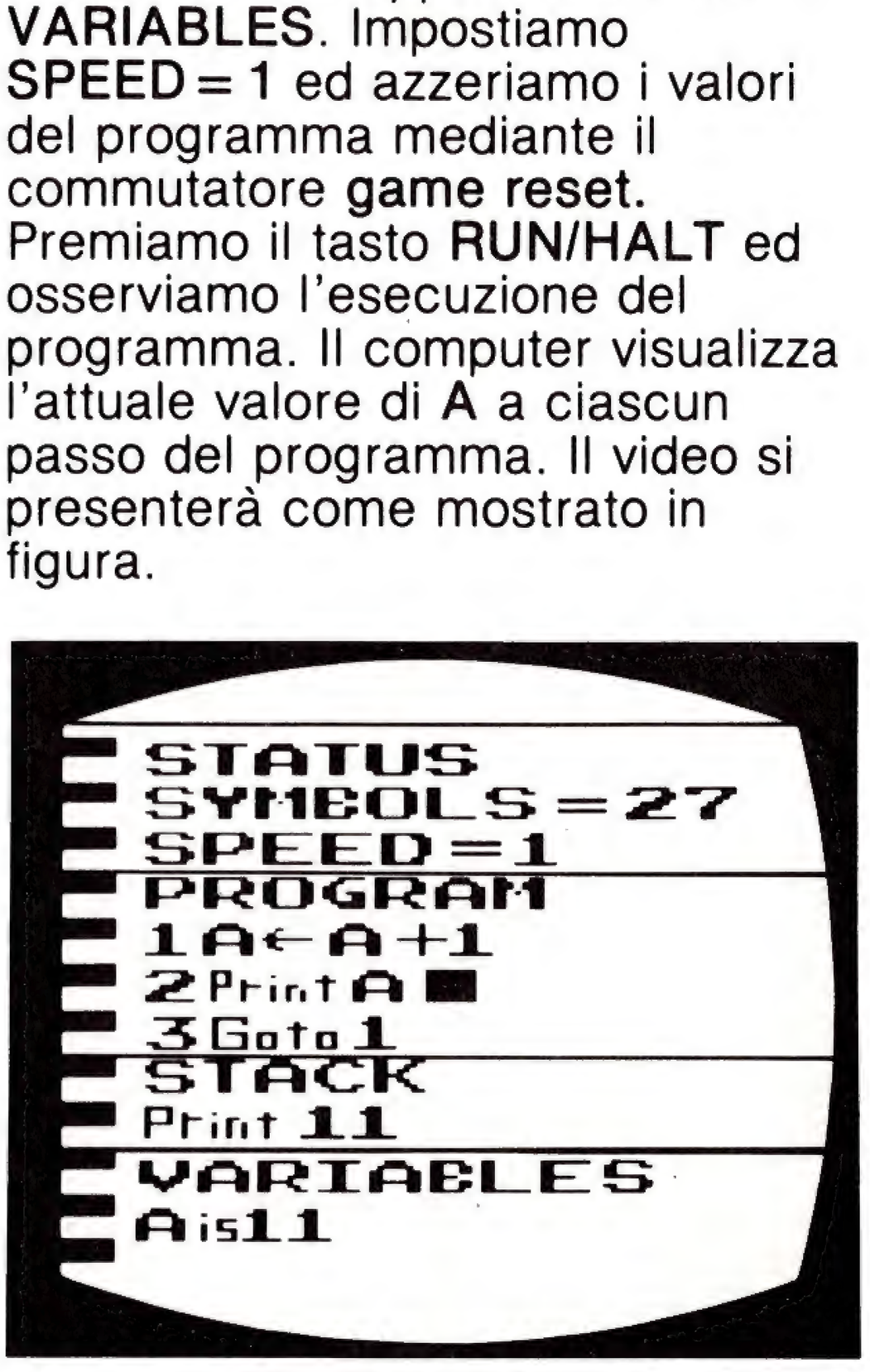
Per arrestare il programma, premiamo **RUN/HALT**. Come si è detto in precedenza, azionando il commutatore **game reset** si cancellano tutti i valori impostati nel programma (senza cancellare però quest'ultimo) e si riporta il medesimo all'inizio.

Analizziamo ora il programma passo per passo mentre viene eseguito dal computer. Portiamo **SPEED** a **60**. Con il cursore nel modo bianco, premiamo il tasto **STEP**. Ogni volta che premiamo questo tasto, viene eseguito un passo di ciascuna parte del programma.

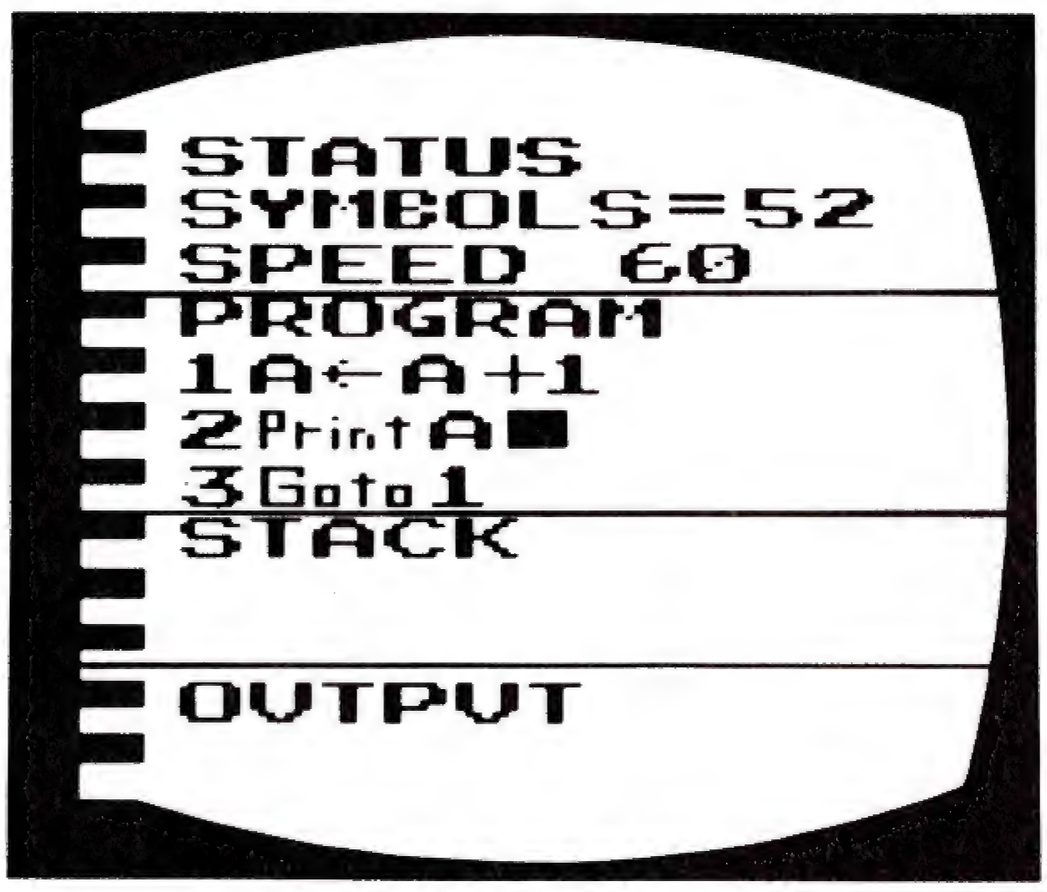
Nella linea **1** il programma è: **A ← A + 1**. Il computer lo legge lo legge come: **A** "diventa" **A + 1**. Osserviamo la zona **STACK** mentre viene eseguita la linea **1**. Nella linea **2** abbiamo detto al computer: **Print A**. Ciò significa che vogliamo che esso mostri il valore di **A** (risultante dalla linea **1**) nella zona **OUTPUT**.

Vedremo in seguito come ciò avviene. La linea **2** dice al computer: **Goto 1**. Il computer ritorna alla linea **1** e trova un nuovo valore di **A**. Ogni volta che si esegue il programma, il computer trova un nuovo valore di **A**, lo visualizza e ritorna quindi alla linea **1**. Questo procedimento continua fino a quando non si esaurisce completamente la "memoria". La quantità di memoria restante è indicata nel settore **SYBMOLS** della zona **STATUS**.

Facciamo ora apparire la zona **VARIABLES**. Impostiamo **SPEED = 1** ed azzeriamo i valori del programma mediante il commutatore **game reset.** Premiamo il tasto **RUN/HALT** ed osserviamo l'esecuzione del programma. Il computer visualizza l'attuale valore di **A** a ciascun passo del programma. Il video si presenterà come mostrato in figura.



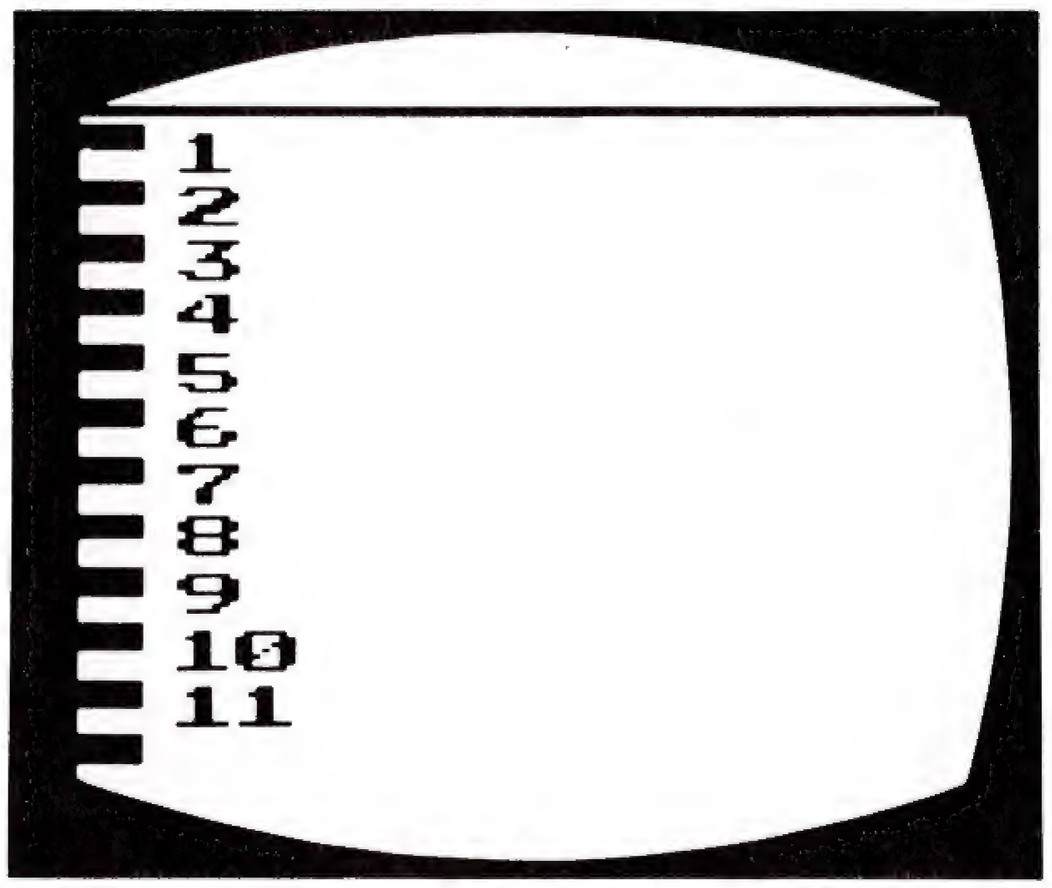
Arrestiamo il programma ed azzeriamo tutti i valori azionando **game reset**. Eliminiamo la zona **VARIABLES** e facciamo comparire la zona **OUTPUT**. Portiamo la velocità a **SPEED = 60**. Il video risulterà come mostrato a fianco.

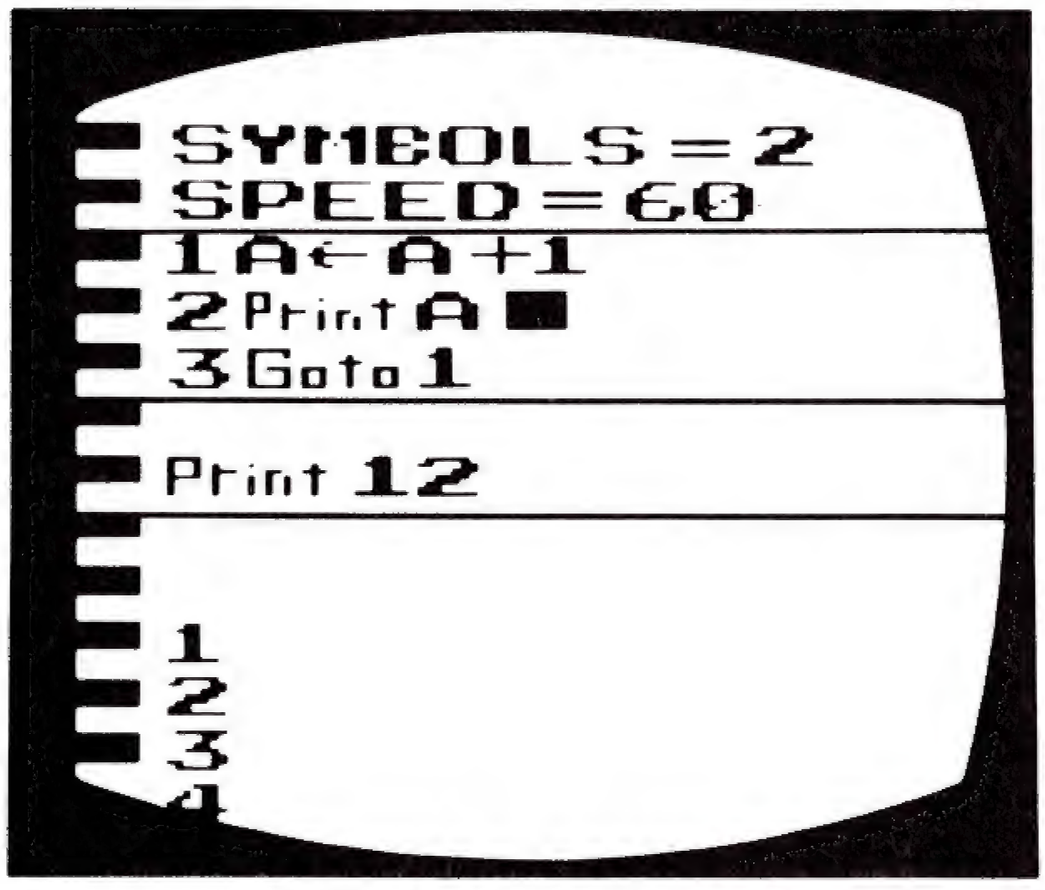


Premiamo **RUN/HALT** e cominciamo il programma. Nella linea **2**, abbiamo comandato al computer: **PRINT A.** Quando arriva a quel punto del programma, il computer visualizza il valore attuale di **A** nella zona **OUTPUT**. Osserviamo il settore **SYMBOLS** della zona **STATUS**.

Anche se il numero **1** appare nella zona **OUTPUT** delvideo, il computer visualizza sempre il nuovo valore di **A**. Portiamo il commutatore **left difficulty** nella posizione **a**. Si otterrà così la presentazione mostrata in figura.

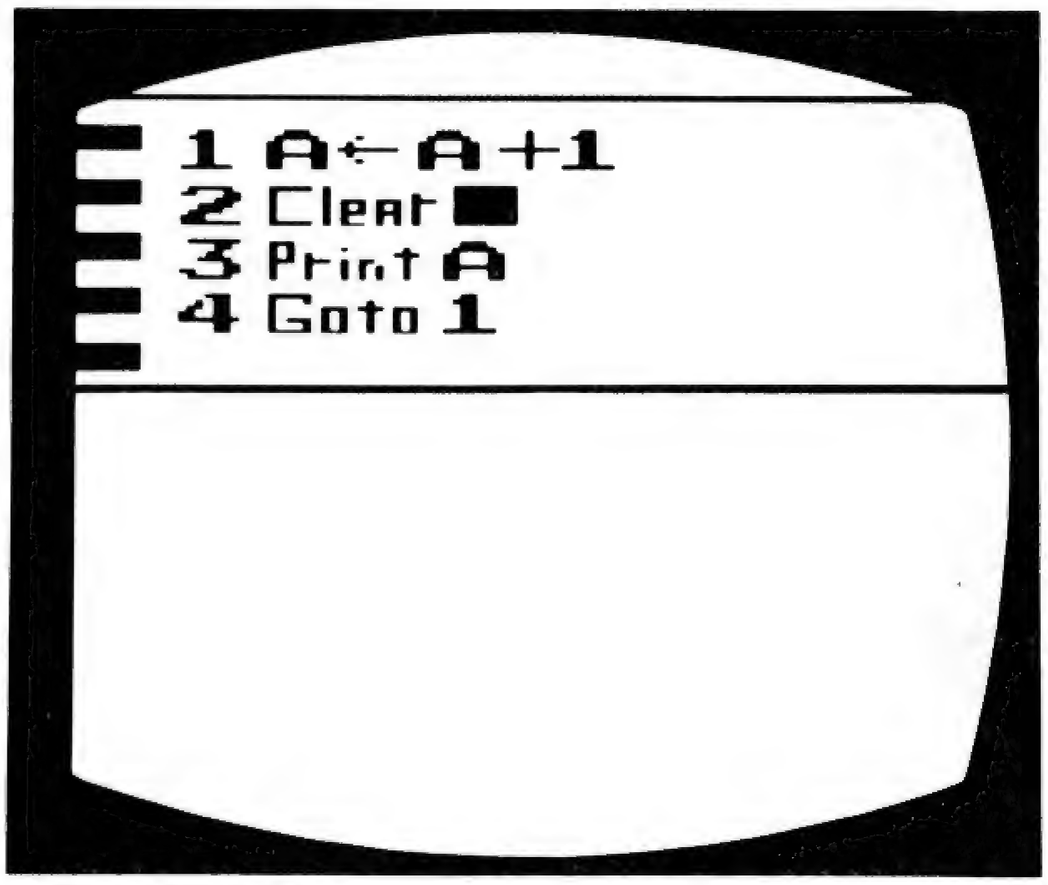
Come fare se vogliamo ampliare la zona **OUTPUT** del video? Si eliminano le zone **STATUS, PROGRAM** e **STACK**, ottenendo la presentazione mostrata a fianco:





Supponiamo di non voler "sovraccaricare" la zona **OUTPUT**. In tal caso, arrestiamo il programma e cancelliamo i valori mediante il commutatore **game reset**. Eliminiamo quindi la zona **OUTPUT** e facciamo apparire la zona **PROGRAM**. Spostiamo il cursore mediante il tasto **FORWARD** al termine della linea **1**, in modo che il video risulti come mostrato in figura.

Premiamo ora il tasto **NEW LINE** ed immettiamo un comando in questo punto, cioè **CLEAR** (modo verde). Otterremo la presentazione mostrata a fianco.



Richiamiamo ora le zone **STACK**, **VARIABLES** e **OUTPUT**. Lasciamo il commutatore **left difficulty** nella posizione **a** e diamo inizio al programma. Oltrepassata la linea **2** CLEAR, viene cancellato il valore di **A** nella zona **OUTPUT** e quindi visualizzato il valore successivo di **A** nella linea **3. BASIC PROGRAMMING** può essere utilizzato soltanto con numeri di due cifre; pertanto, una volta raggiunto il numero **99**, il programma si "avvolge su sè stesso" e ricomincia mostrando un valore di **A** pari a **0**.

USO DELLA FUNZIONE NOTE

Ogni volta che il programma immagazzina un numero nel modo **NOTE** (rosso), l'altoparlante della televisione emette una nota musicale.

Prepariamo alcuni programmi a scopo dimostrativo. Se nel Video Computer System ci sono precedenti programmi, azioniamo il commutatore **game select** della console.

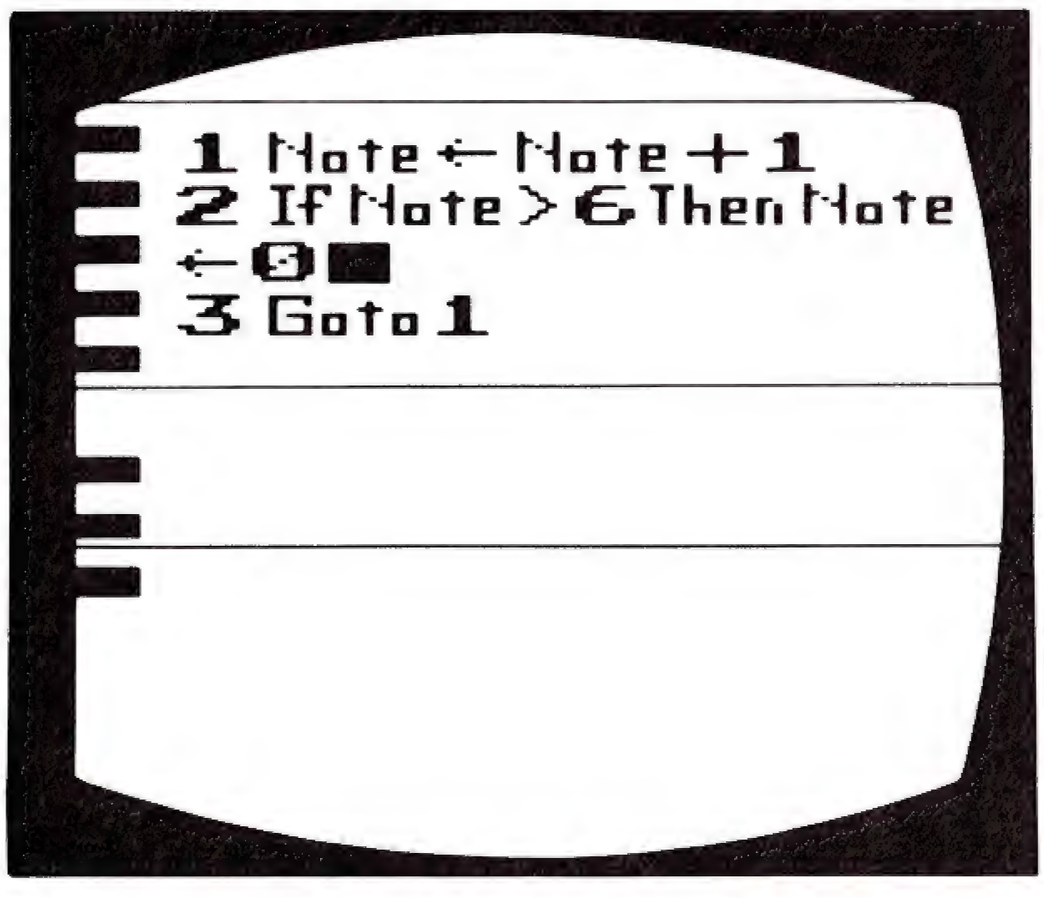
Impostiamo quanto segue:

```
1  Note ← Note + 1
2  Goto 1
```

Diamo inizio quindi al programma. Rallentiamo la velocità di esecuzione ed osserviamo come il programma viene visualizzato nella zona **STACK**. Notiamo come il valore attuale di **NOTE** viene mostrato nella zona **VARIABLES**. Arrestiamo il programma ed inseriamo una nuova linea **2** (nel modo bianco, porteremo il cursore al termine della linea **1** e premeremo il tasto **NEW LINE**. In tal modo la linea **2** diventa linea **3**).

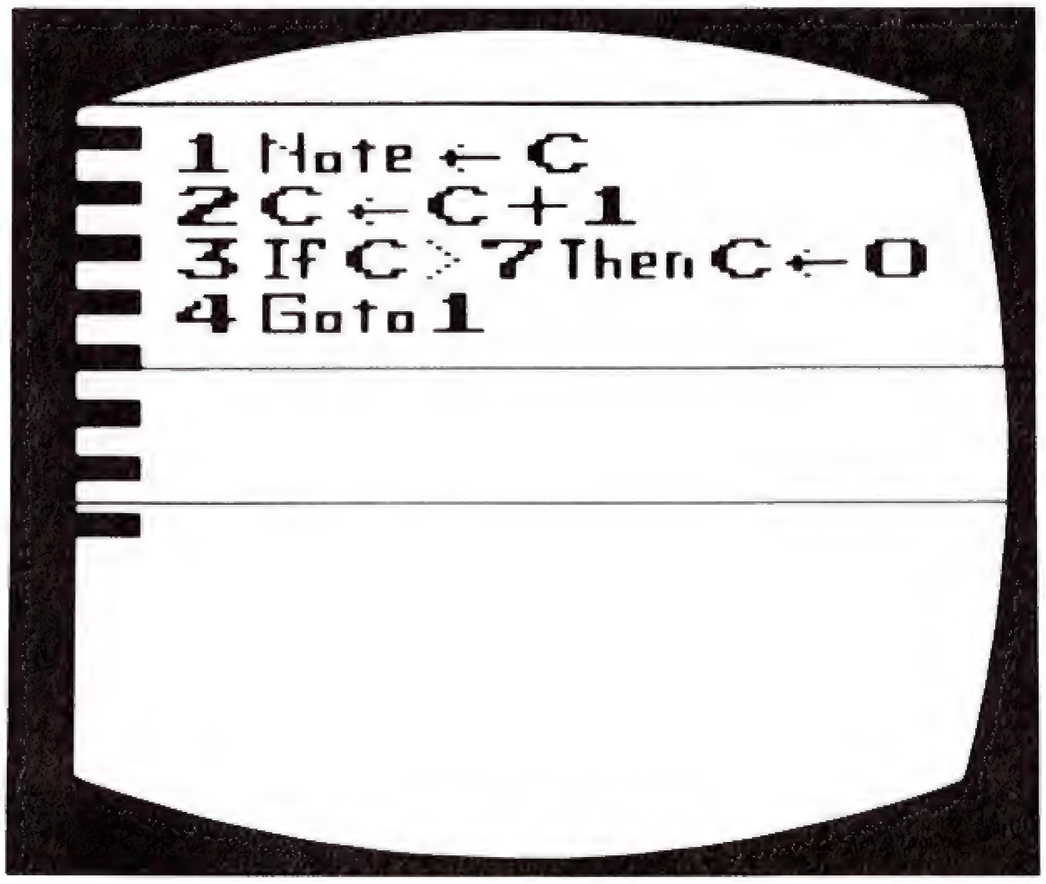
```
2  If Note > 6 Then Note ← 0.
```

Il video risulterà come mostrato in figura (qualora vengano eliminate le zone **STACK** e **VARIABLES**).



Con i comandi **IF** e **THEN** diciamo al computer che SE (**IF**) succede una data cosa, ALLORA (**THEN**) esso dovrà eseguire qualche altra cosa. Nell'esempio, IF Note è maggiore di **6, THEN** Note deve essere portata a **0**. Cominciamo il programma ed osserviamo come viene visualizzato nella zona **STACK**. Notiamo che quando il valore di Note raggiunge **7**, esso diventa maggiore di (>) **6** e quindi il programma porta il valore a **0**.

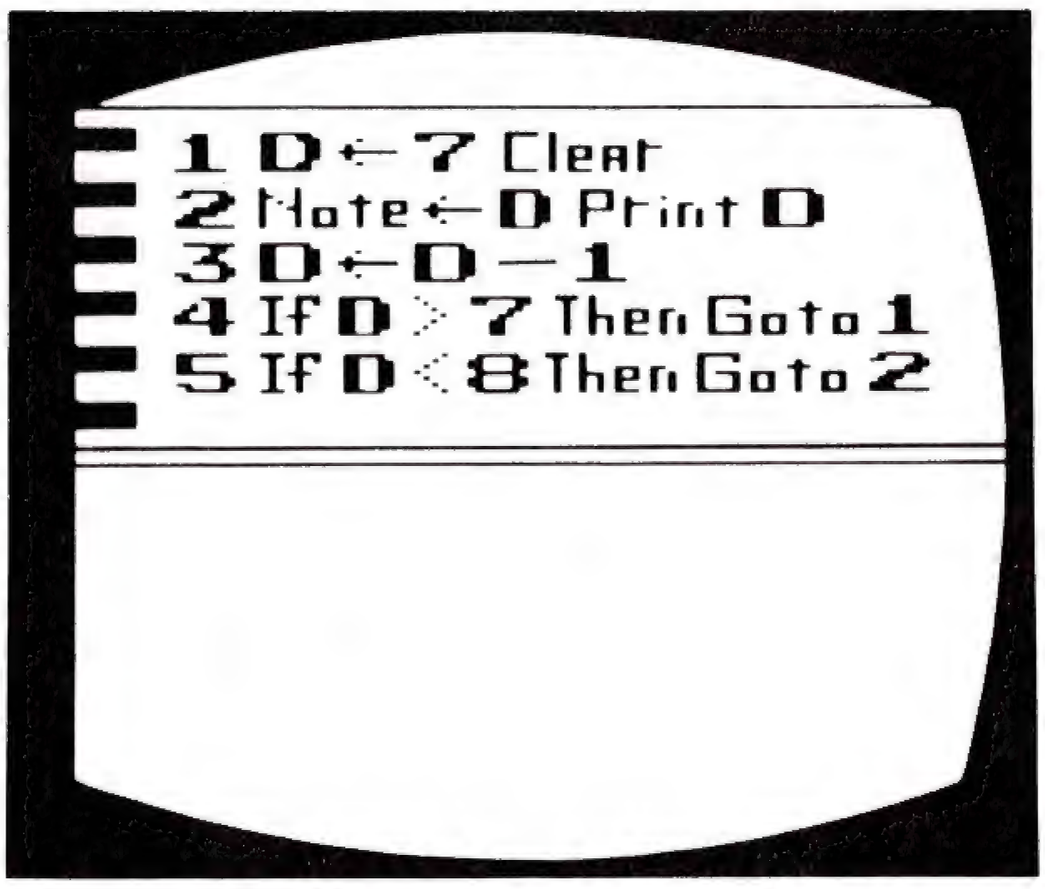
Elaboriamo un altro programma per ottenere lo stesso risultato in un diverso modo. Impostiamo quanto segue:



Facciamo eseguire il programma dal computer. Notiamo che i risultati ottenuti sono identici, ad eccezione del fatto che le note sono riprodotte ad intervalli uguali. Se vogliamo che il valore di **C** sia visualizzato nella zona **OUTPUT**, impostiamo **Print C** e **Clear** in qualsiasi punto del programma. Per ridurre il tremolio nella zona

OUTPUT quando il programma dà il valore suddetto, inseriamo una virgola (modo verde) dopo di esso nella linea **Print, Print C**.

Per le variabili del programma si può usare qualsiasi lettera dell'alfabeto. Il seguente programma comprende l'uso di tutti i tasti finora spiegati ed utilizza quasi completamente la "memoria" disponibile nel computer. Dopo aver impostato questo programma, eliminarlo dal video e richiamare le zone **STATUS, STACK** ed **OUTPUT**, eseguendo infine il programma.



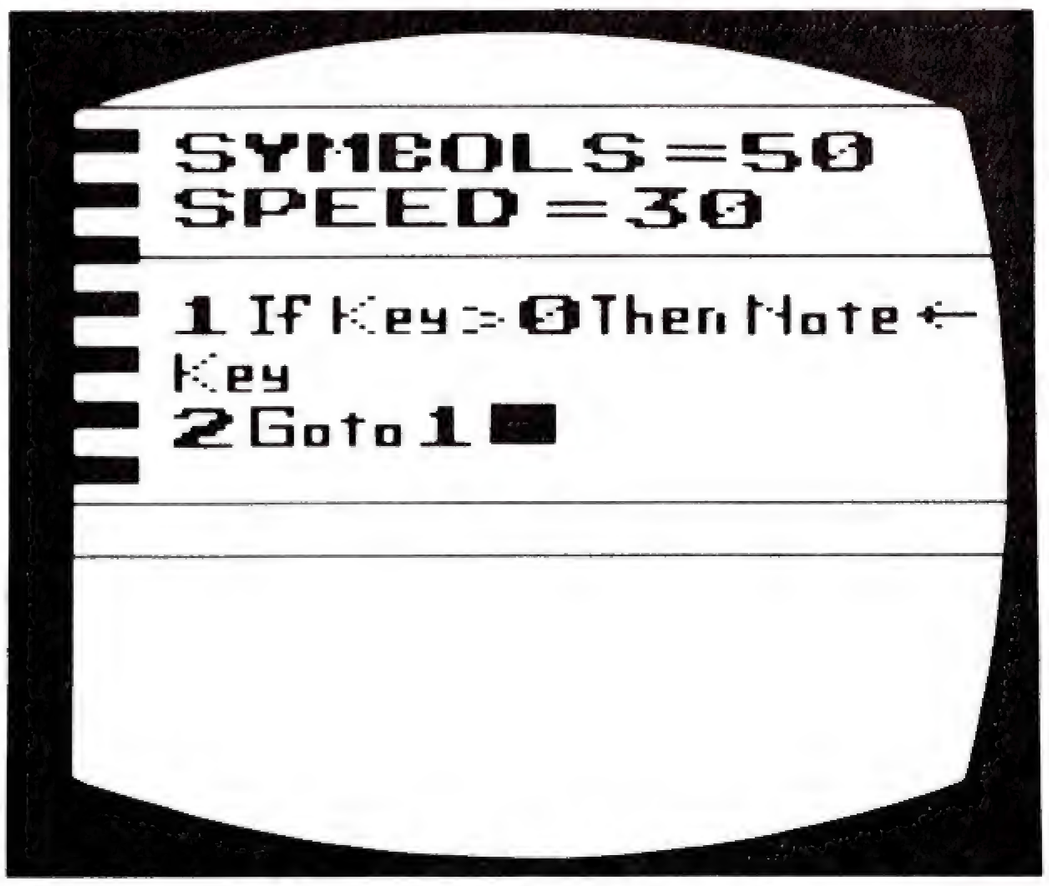
In questo programma abbiamo impostato due comandi **IF/THEN**. Se le condizioni di cui al primo comando non vengono realizzate (linea **4**), il computer passa alla linea successiva (linea **5**). Se la memoria è sufficiente, si possono impostare vari comandi tra due comandi **IF/THEN**.

In taluni casi possiamo impostare due comandi su una stessa linea, come nel caso della linea **1**e della linea **2** nel programma di cui sopra. In tal modo si risparmia memoria per una parte successiva del programma.

FUNZIONI KEY E PRINT

La funzione **KEY** (modo rosso) serve ad immettere una variabile nel programma mentere questo viene eseguito dal computer. Il programma elabora in tal caso **KEY** e lo sostituisce con un numero impostato mediante i tasti sul lato destro del comando a tasti. Se non viene immesso nessun numero, il programma legge **KEY** come se fosse **0**.

Il seguente è un semplice esempio di programma in cui viene utilizzata la funzione **KEY**:



Diamo inizio al programma e premiamo alcuni dei testi sul lato destro del comando. Con un po' di pratica riusciremo ad eseguire un motivo musicale.

Come vedremo in seguito, la funzione **KEY** può essere utilizzata anche per programmare nella zona **GRAPHICS**.

Proviamo questo programma usando le funzioni **KEY, NOTE** e **PRINT**:

```
SYMBOLS=34
SPEED=30
1 A←Key
2 Note←A Print A
3 If A=0 Then Clear
4 Goto 1
9
```

Osserviamo il programma nella zona **OUTPUT**. Notiamo che quando impostiamo un numero mediante un tasto sul lato destro del comando, il programma esegue la nota corrispondente e visualizza il numero nella zona **OUTPUT**.

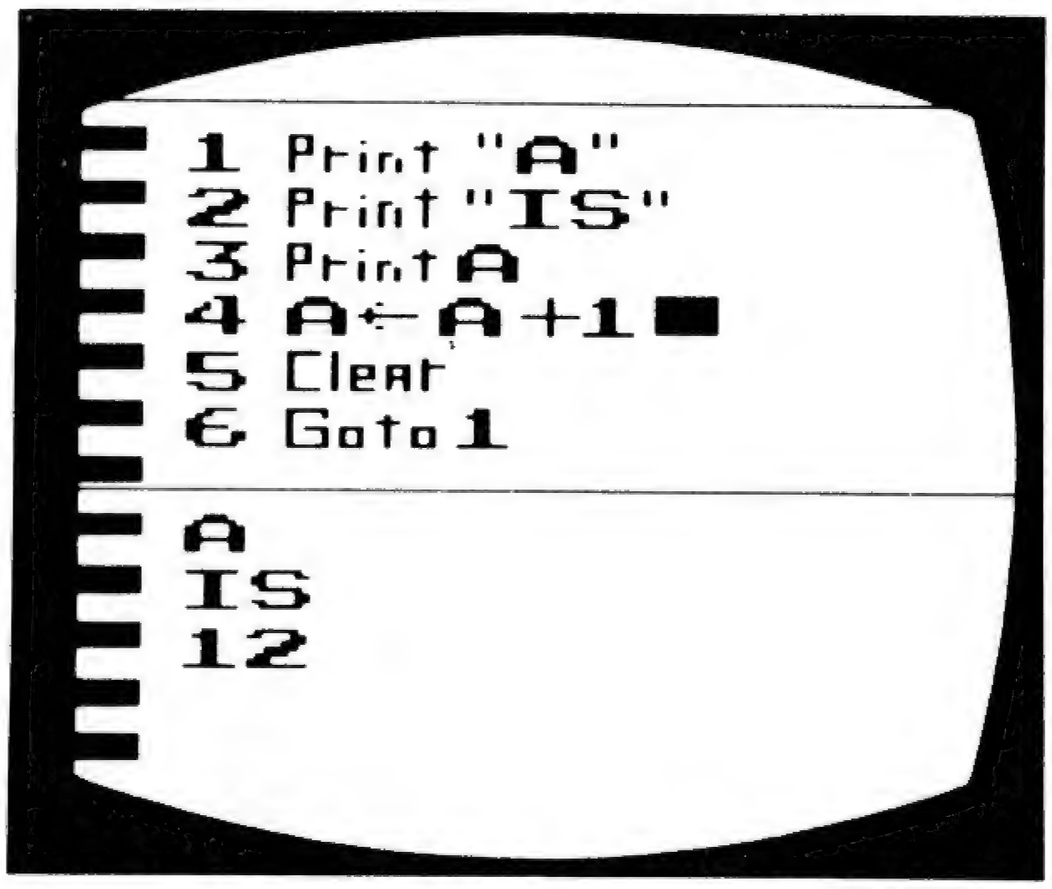
Modifichiamo lievemente il programma. Inseriamo una virgola (,) nella linea **2** dopo **Print A** e diamo inizio al programma. Inserendo la virgola in quel punto abbiamo detto al computer che vogliamo la visualizzazione del maggior numero possibile di variabili in una stessa linea.

FUNZIONE PRINT

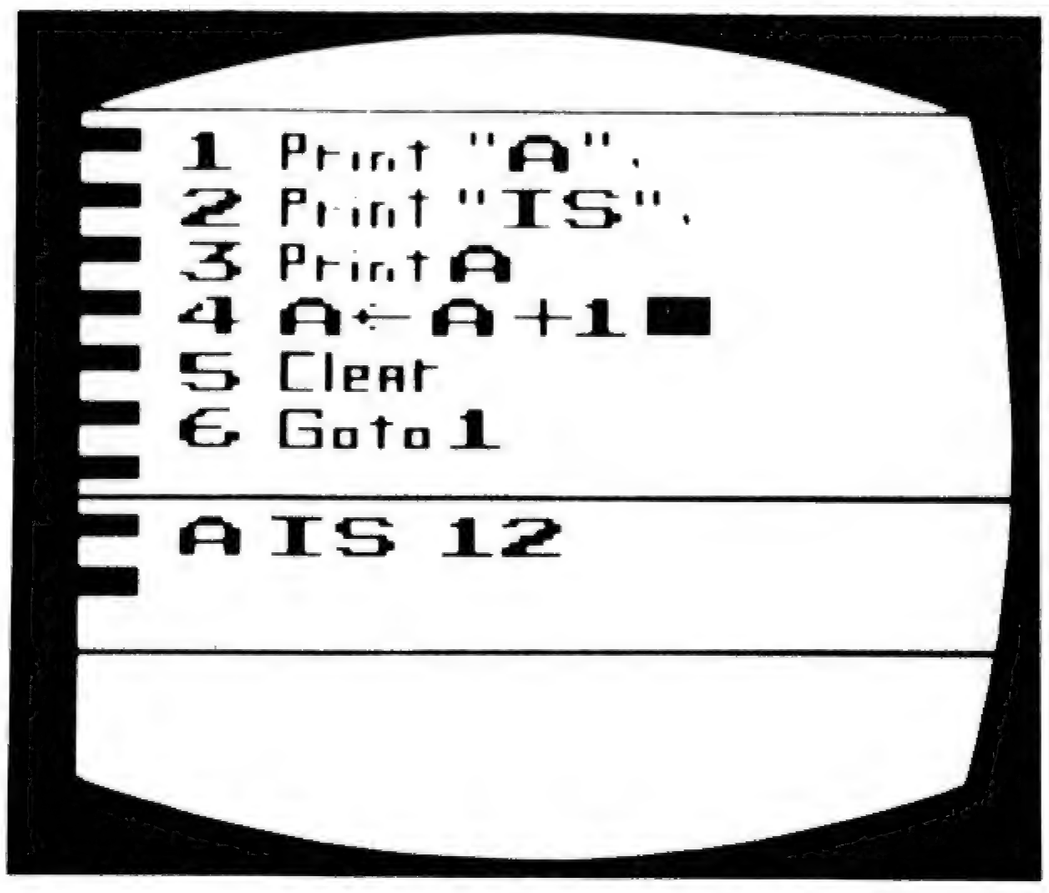
Come si è visto in precedenti programmi, la funzione **PRINT** può essere utilizzata per dire al computer di visualizzare il valore di una data variabile nella zona **OUTPUT**. La funzione serve inoltre a far visualizzare dal programma determinate parole. Tali parole devono essere racchiuse tra virgolette (''). Impostiamo **SPEED = 8** ed immettiamo il seguente programma:

```
1 Print "A"
2 Print "IS"
3 Print A
4 A←A+1
5 Clear
6 Goto 1
```

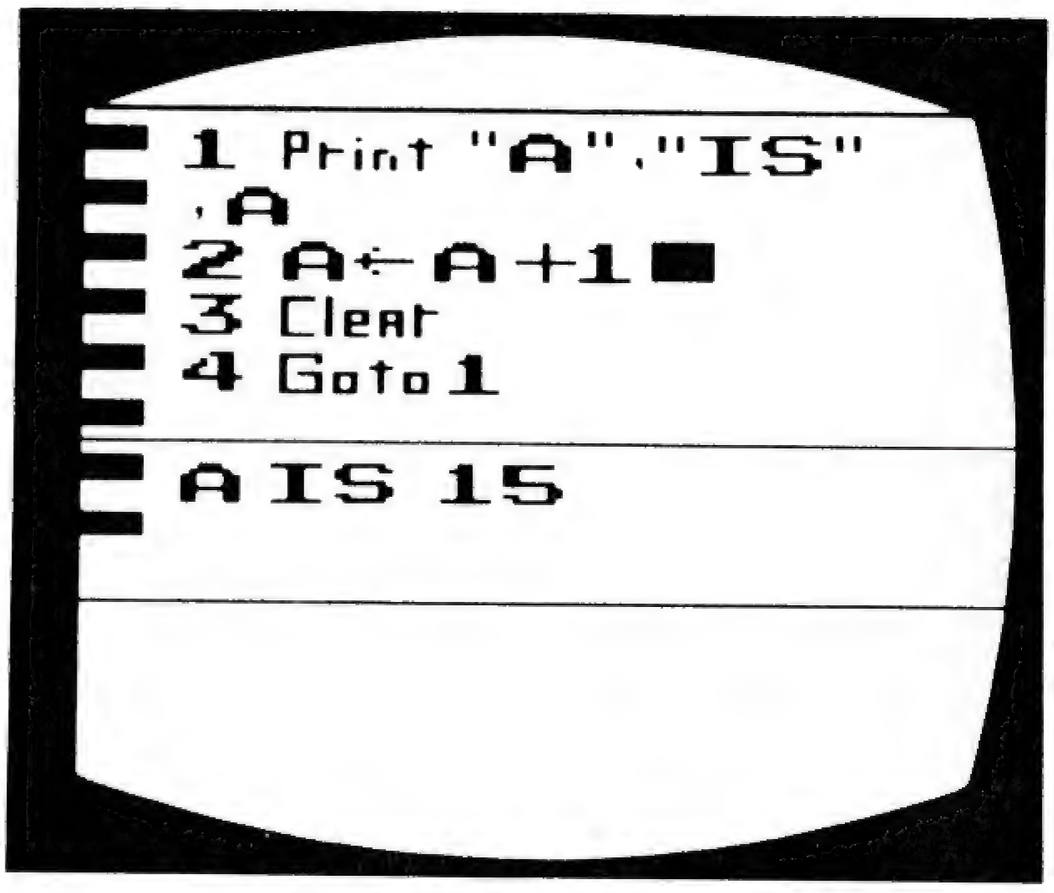
Eseguiamo ora il programma ed osserviamo cosa succede nella zona **OUTPUT**. Il programma visualizza le parole impostate, ma queste sono ''accatastate'' (**stacked**).



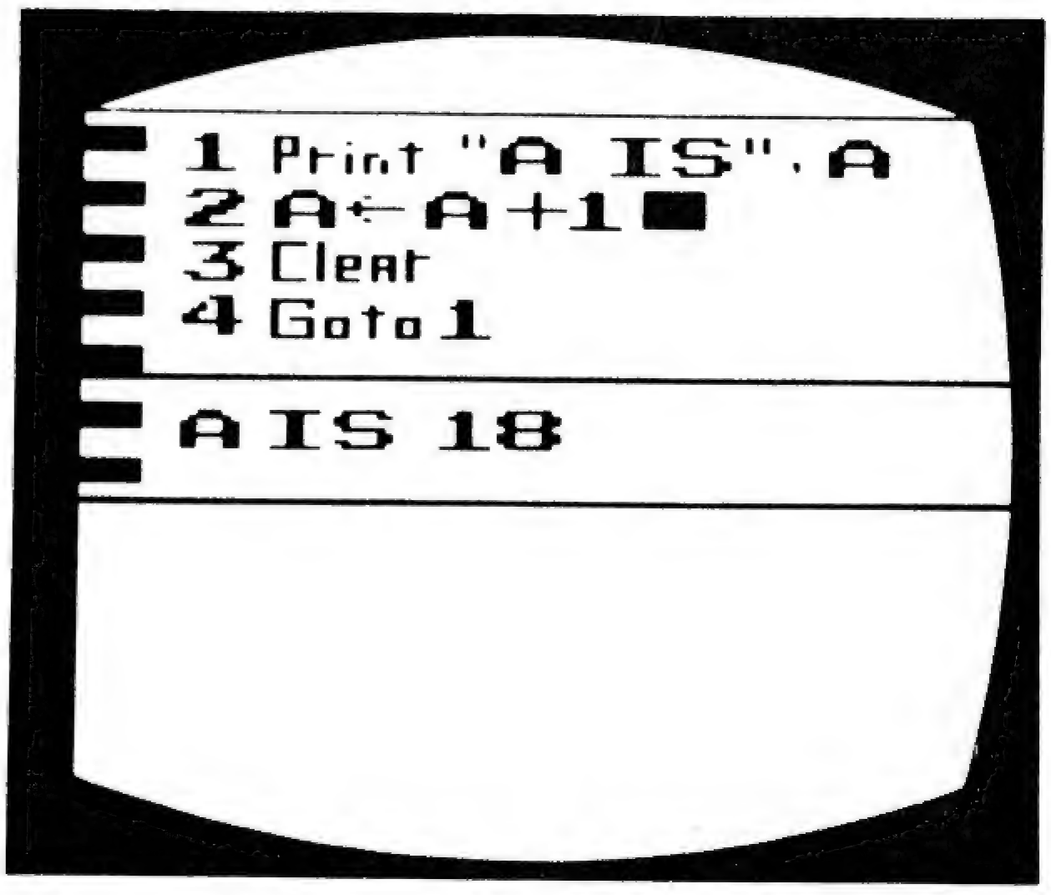
Modifichiamo ancora leggermente il programma inserendo una virgola (,) al termine della linea **1** e della linea **2**. Il programma si presenta ora così:



Mediante la virgola (,) impostata nella linea **1** abbiamo detto al computer che tutto ciò che si trova nella linea **2** deve essere visualizzato nella zona **OUTPUT** nella stessa linea dell'istruzione contenuta nella linea **1**. Con la virgola nella linea **2** abbiamo detto al computer che l'istruzione di cui alla linea **3** deve seguire la linea **2**. Le suddette istruzioni possono essere abbreviate modificando il programma nel modo seguente:



Il programma può essere ulteriormente abbreviato come segue:



Impostando il programma in questo modo consente di risparmiare memoria.

UTILIZZAZIONE DELLA ZONA GRAPHICS

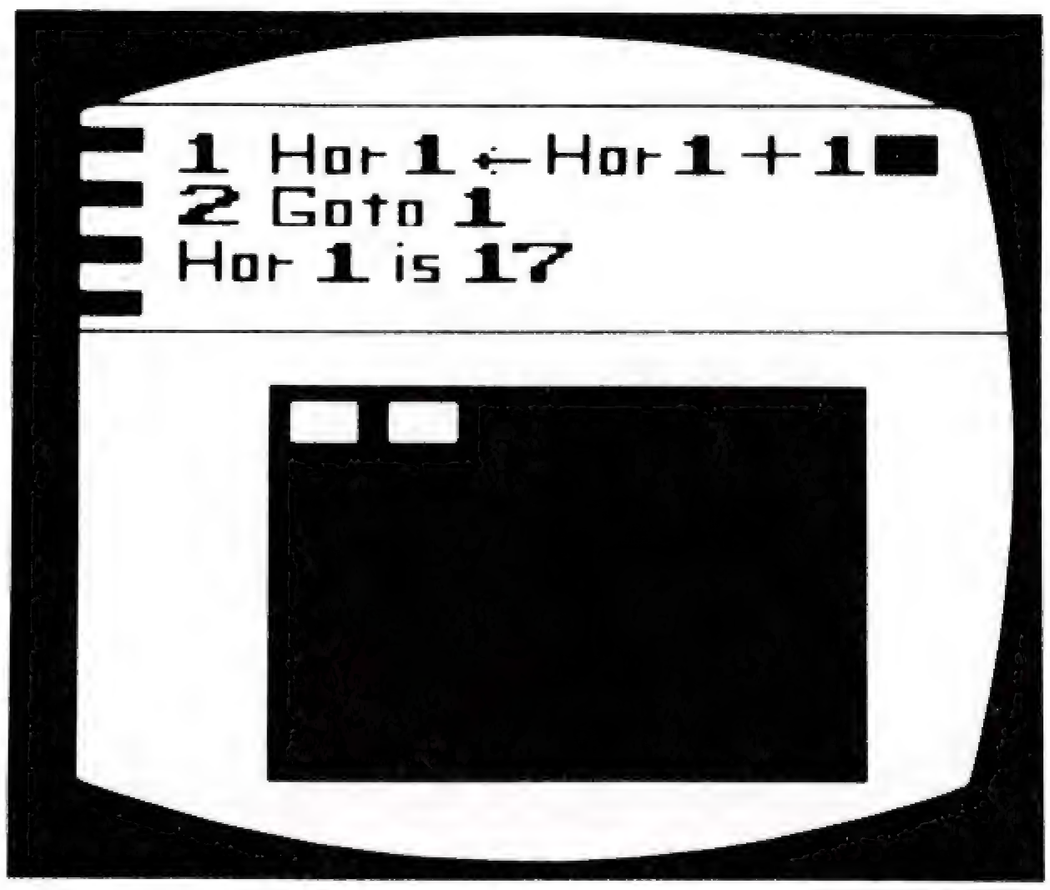
La zona **GRAPHICS** è delimitata dal rettangolo blu del video. A prima vista essa sembra contenere un rettangolino rosso nell'angolo superiore sinistro; in realtà, tale rettangolino ne copre uno bianco ed entrambi possono essere spostati uno indipendentemente dall'altro nel campo governato dal programma.

Per spostare i rettangolini è necessario cambiarne le coordinate. Il rettangolino rosso è l'oggetto numero **1**; la sua coordinata orizzontale è governata da **Hor 1** sul comando a tasti, quella verticale da **Ver 1**. Le coordinate dell'oggetto numero **2**, cioè il rettangolino bianco, sono impostate mediante **Hor 2** e **Ver 2**.

Entrambi i rettangolini od oggetti si trovano inizialmente nell'angolo superiore sinistro del campo blu, ma si vede soltanto quello rosso. Tale angolo rappresenta la posizione zero (**0**) od origine. Quando alle variabili **Hor 1, Ver 1, Hor 2** e **Ver 2** non viene assegnato un particolare valore, il loro valore è **0** e pertanto i due rettangolini restano nell'angolo superiore sinistro.

Quando ad una coordinata è assegnato un valore diverso da 0 (es.: **Hor 1 ← 10**), l'oggetto in questione salta nella posizione corrispondente a tale valore nel campo blu quando si esegue il programma.

Impostiamo il seguente programma:

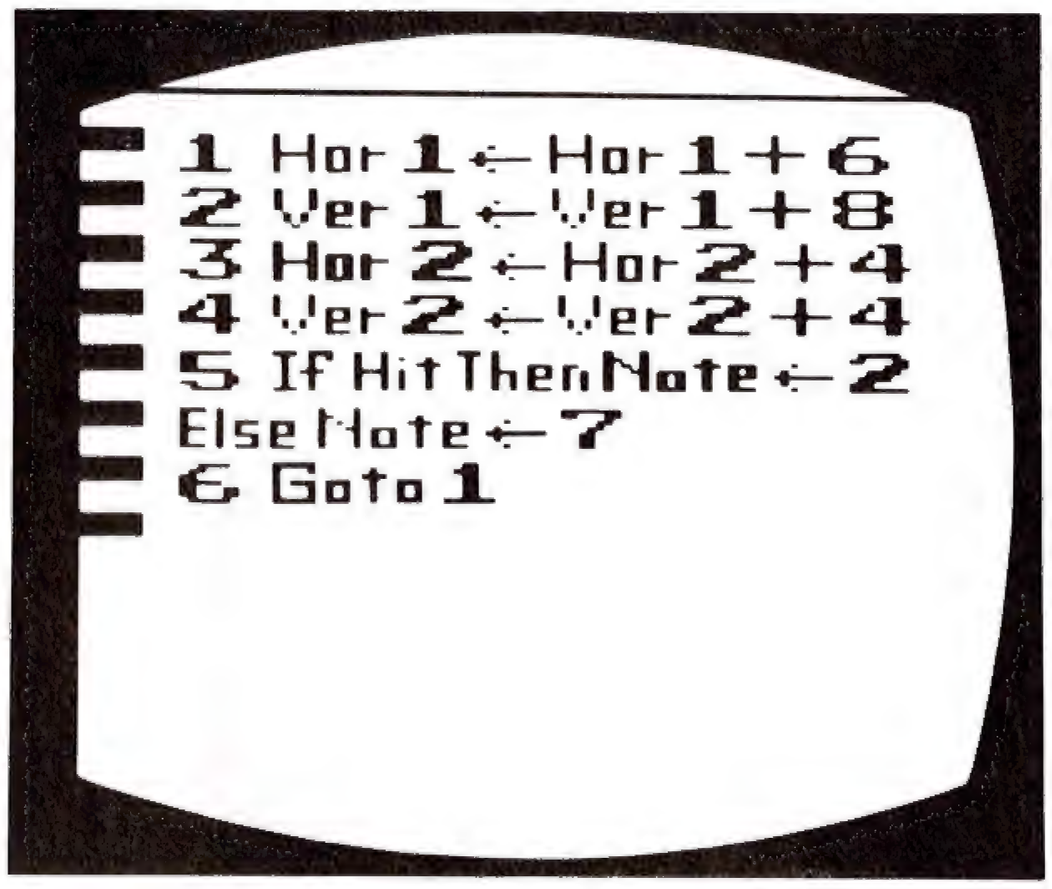


Questo programma comporta lo spostamento del rettangolino rosso a destra, uno spazio orizzontale alla volta. Quando si dà inizio al programma si osserva il lento spostamento di questo rettangolino verso il lato destro del video. Controlliamo la zona **VARIABLES** ed osserviamo come il valore di **Hor 1** vada aumentando. Una volta raggiunto il valore di **99**, cioè il massimo consentito, **Hor 1** ritorna a **0**. Quando tale coordinata raggiunge il valore di **99**, il rettangolino rosso avrà raggiunto l'estremità destra del campo blu. Dopodiché, esso si "riavvolge", cioè ricompare all'estremità sinistra del campo, con un volore di **Hor 1** pari a **0**.

Le coordinate orizzontali (**Hor 1** e **Hor 2**) hanno pertanto un valore compreso tra **0** e **99**, come anche le coordinate verticali (**Ver 1** e **Ver**

2). Il valore **0** corrisponde alla cima del campo blu e il **99** al fondo.

Impostiamo il seguente programma:



Eliminiamo il programma dal video ed assicuriamoci che la zona **GRAPHICS** sia interamente visibile. Impostiamo una velocità pari a **60** ed eseguiamo il programma. Esso mostra una particolare modo di spostare i rettangolini da un punto all'altro del campo, nonché alcune utilizzazioni delle funzioni **HIT** ed **ELSE**.

Nella linea **5** il programma dice al computer di suonare la nota **2 IF** (SE) i rettangolini **HIT** (entrano in collisione), il che si verifica di tanto in tanto. **HIT** significa che i rettangolini devono avere le stesse coordinate. Diversamente (**ELSE**) il computer dovrà suonare la nota **7**, e ciò si verificherà fino a quando i rettangolini entrano in collisione.

(Togliendo **ELSE NOTE ← 7** dalla linea **5**, quando i rettangolini entrano in collisione verrà eseguita esclusivamente la nota **2**).

FUNZIONE MOD

Mod è un operatore aritmetico, molto simile alla divisione (÷). BASIC PROGRAMMING esegue divisioni con quoziente intero, senza indicazione del resto. Ad esempio, 14÷5 è uguale a 2 e 4/5, ovvero a 2 con resto di 4; nel BASIC PROGRAMMING, invece, il risultato è 2. In altre parole, il 5 sta nel 14 2 volte (2 x 5 = 10) ed avanza 4; BASIC usa invece esclusivamente numeri interi, e quindi la risposta è 2.

Il computer dà quindi la stessa risposta sia per 12 ÷ 4 che per 13 ÷ 4. In entrambi i casi il risultato è 3, perché il 4 sta 3 volte sole tanto nel 12 quanto nel 13. Per la seconda operazione, il computer non riconosce che vi è un resto di 1.

Tornando alla funzione **Mod**, essa calcola ed indica il resto della divisione del primo numero impostato per il secondo. Pertanto, 14 **Mod 5** è uguale a **4**. Il **5** sta nel **14** due volte (**2 × 5 = 10**) ed avanza 4. **Mod** calcola e dà il resto, quindi **Mod = 4.**

Quanto fa **13 Mod 4?** La risposta è **1**, cioè il resto ottenuto dividendo **13** per **4 (3 × 4 = 12)**. E **12 Mod 4?** Qui il risultato è **0** perché il **4** sta nel **12** esattamente **3** volte senza resto.

In genere, dal punto di vista strettamente matematico, la divisione di un numero per **0** è indeterminata. Nel BASIC PROGRAMMING dividendo per **0** si ottiene **0**, per cui **5 ÷ 0 = 0.**

ORDINE DI PRIORITÀ DEGLI OPERATORI

In un'espressione matematica, le operazioni arithmetiche vengono eseguite secondo un determinato ordine di priorità assegnato ai relativi operatori (+ , —, x, ÷ , Mod, etc.). BASIC PROGRAMMING adotta il seguente ordine:

1. x ÷
2. + —
3. Mod
4. =
5. ←

USO DELLE PARENTESI

Le parentesi (modo verde, lato destro del comando a tasti) conferiscono la precedenza ai numeri in esse racchiusi ai fini dello svolgimento di un'espressione.

Esempio:

A 5 + 3 x 2 Mod 7
primo operazione: 3 x 2 = 6
seconda operazione: 5 + 6 = 11
terza operazione: 11 Mod 7 = 4
A = 4

Inserendo le parentesi nella stessa espressione si ha un diverso svolgimento ed un diverso risultato finale:

A (5 + 3) x 2 Mod 7
prima operazione: 5 + 3 = 8
seconda operazione: 8 x 2 = 16
terza operazione: 16 Mod 7 = 2
A = 2

ESEMPI DI PROGRAMMI

I seguenti esempi di programma serviranno a dare un'idea dell'ampio arco di possibilità che caratterizza la programmazione. È necessario tenere a mente gli effetti prodotti dall'uso di determinate funzioni (**IF, THEN, ELSE, PRINT, KEY** etc.) sul particolare programma elaborato. Si ricordi inoltre che la funzione **KEY** è legata ai comandi impostati mediante i tasti di destra per quanto riguarda la corretta esecuzione del programa da parte del computer.

Se un determinato programma crea delle perplessità, arrestarlo, riportarlo al punto di partenza ed eseguirlo passo per passo

azionando lo **STEP**. Osservando in tal modo il programma nella zona **STACK**, si sarà in grado di comprendere cosa sta succedendo e perché.

Dopo aver eseguito tali programmi ed avere assimilato i principi fondamentali della programmazione in generale, si potrà cominciare ad affrontare la realizzazione di programmi individuali.

HIT

```
1 Clear PrintHit
2 Hor 2 ← Hor 2 + 1 Mod
3 3
3 Goto 1
```

PROGRAMMA MUSICALE

```
1 Note ← If Note > 4
Then Note — 11 Else
Note + 1
2 Note ← Note + 2
3 Note ← Note — 2
4 Goto 1
```

```
1 A ← Key
2 If A > 0 Then Note ← A
— 1
3 Goto 1■
```

```
1 Note ← Note + Key +
7
2 Goto 1■
```

```
1 A ← A + 1
2 Note ← A
3 If (A Mod 2) = 0 Then
Note ← 0 Else Note ← 4
4 Goto 1■
```

PROGRAMMA CLOCK-LIKE ("OROLOGIO")

Per l'esecuzione di questo programma richiamare soltanto la zona **OUTPUT**. Impostare **SPEED = 30** o **60.**

```
1 Clear
2 A ← A + 1 Mod 60
3 If A = 0 Then B ← B + 1
Mod 60
4 Print B. A
5 Goto 1■
```

PROGRAMMA GIOCO PONG®
(PALLINA E RACCHETTA)

```
1 Hor 1 ← Hor 1 + Key
2 Hor 2 ← Hor 2 + 8
3 Ver 2 ← Ver 2 - 3
4 If Hit Then Ver 2 ← 93
. Note ← 7
5 Goto 1
```

PROGRAMMA GIOCO PONG®
(senza effetti sonori)

```
1 Hor 2 ← Hor 2 + Key
2 If Ver 1 > 90 Then
Ver 1 ← 88
3 If Hit Then Ver 1 ← 3
4 Ver 1 ← Ver 1 + If
Ver 1 Mod 2 Then 8 Else
92
5 Hor 1 ← Hor 1 + 7
6 Goto 1■
```

BASIC PROGRAMMING

JUEGO VIDEO

BASIC PROGRAMMING

INTRODUCCIÓN

Este juego instructivo de programación básica (**BASIC PROGRAMMING**) tiene el objeto de enseñar los pasos fundamentales de programación de computadora. La palabra **BASIC** está formada por las iniciales de *Beginners* ***A****ll-purpose* ***S****ymbolic* *Instruction* ***C****ode*, que en castellano significa Código Universal de Instrucciones Simbólicas para Principiantes. Se creó para que la gente aprendiera fácilmente a "escribir" programas para computadoras.

Los programas para computadoras están constituidos por una serie de instrucciones, y controlan el flujo de información en los medios internos de la computadora. El juego de BASIC PROGRAMMING permite ingresar en el Video Computer System™ las instrucciones necesarias para ejecutar tarea sencillas.

Se debe recordar que el juego BASIC PROGRAMMING cuenta con una cantidad limitada de memoria, comparado con sistemas de computación más elaborados. Sin embargo, es un medio instructivo excelente para el parendizaje de los principios básicos de programación de computadoras.

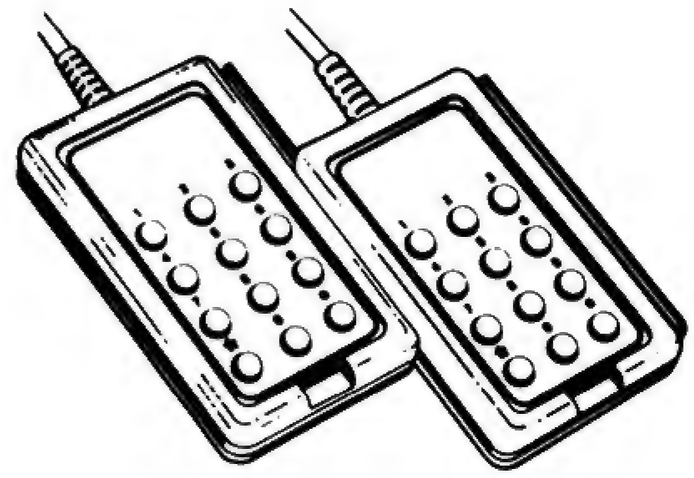
NOTA: Apague la consola antes de insertar o de sacar el cartucho de un programa de juego. No haga excpciones a esta regla. Así protegerá los componentes electrónicos contra posibles daños y prolongará la vida del ATARI® Video Computer System™.

Este programa de juego podría hacer que la imagen en algunas pantallas de televisión se corra verticalmente. En tal caso hay que ajustar el control de estabilización vertical.

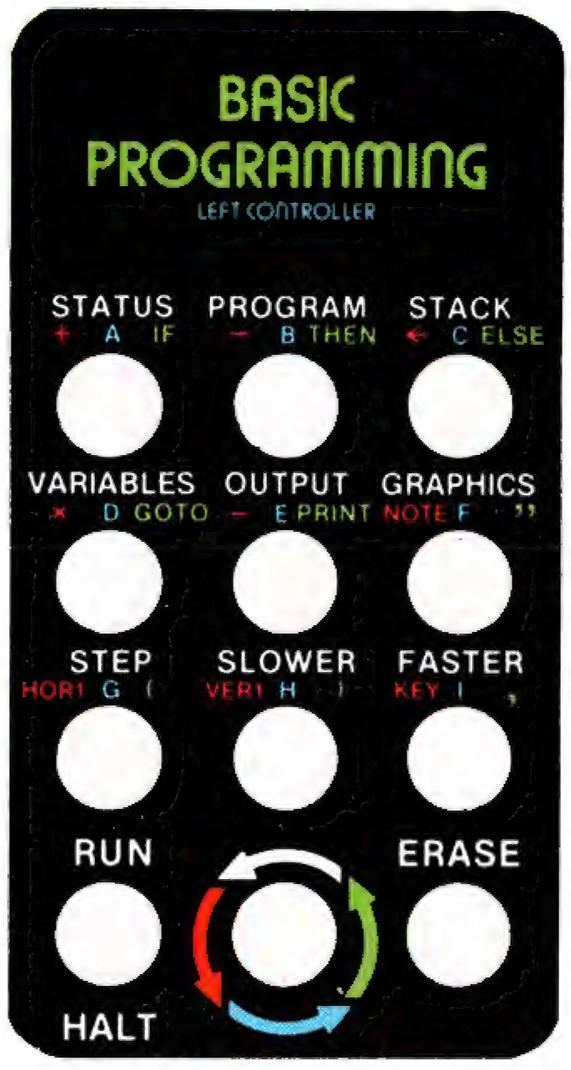
COMANDOS DE TECLADO

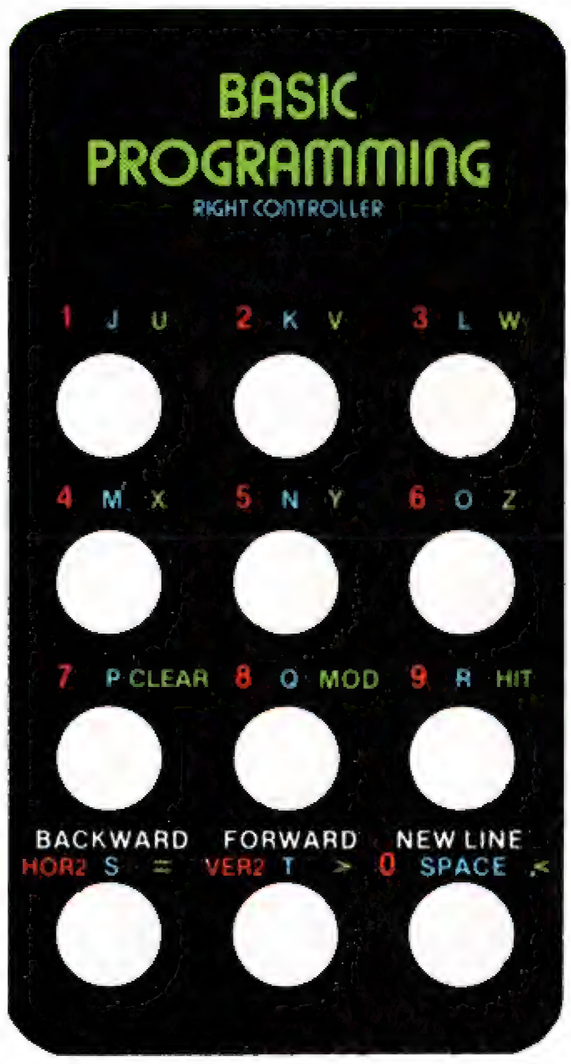
Utilice los comandos de teclado con este cartucho de juego de programación BASIC ATARI® Game Program™. Asegúrese de que el cable conductor de los comandos ha quedado firmemente insertado en el receptáculo de conexión respectivo **LEFT CONTROLLER** y **RIGHT CONTROLLER** en la parte posterior de la consola.

Consulte la Sección 3 del Manual del Usuario si necesita información adicional.



Para conectar los dos comandos de teclado entre sí, deslice la lengüeta del comando izquierdo en la ranura del comando derecho. Los dos comandos unidos de este modo formarán un teclado de 24





teclas que se usará para ingresar los datos de los programas y para controlar la presentación en la pantalla de televisión.

Saque las etiquetas de control de teclado que se encuentran en el sobre. Coloque la etiqueta marcada **LEFT** (izquierda) sobre el comando enchufado en el receptáculo de conexión marcado **LEFT CONTROLLER** en la parte posterior de la consola. Coloque la etiqueta marcada **RIGHT** (derecha) sobre el comando enchufado en el receptáculo de conexión marcado **RIGHT CONTROLLER**.

SECTORES DE LA PANTALLA

La pantalla está dividida en seis sectores:

1. El sector de programación **PROGRAM** se utiliza para ingresar en la pantalla las instrucctiones del jugador.

2. El sector de la escala **STACK** presentará los resultados parciales del programa durante su ejecución.

3. El sector de las varibales **VARIABLES** presentará el valor de cada variable del programa durante su ejecución.

4. El sector de salida **OUTPUT** presentará los datos de salida que se van produciendo durante la ejecución del programa.

5. El sector de estado interno **STATUS** de la máquina muestra la cantidad de memoria disponible en cualquier momento durante la ejecución del programa. En

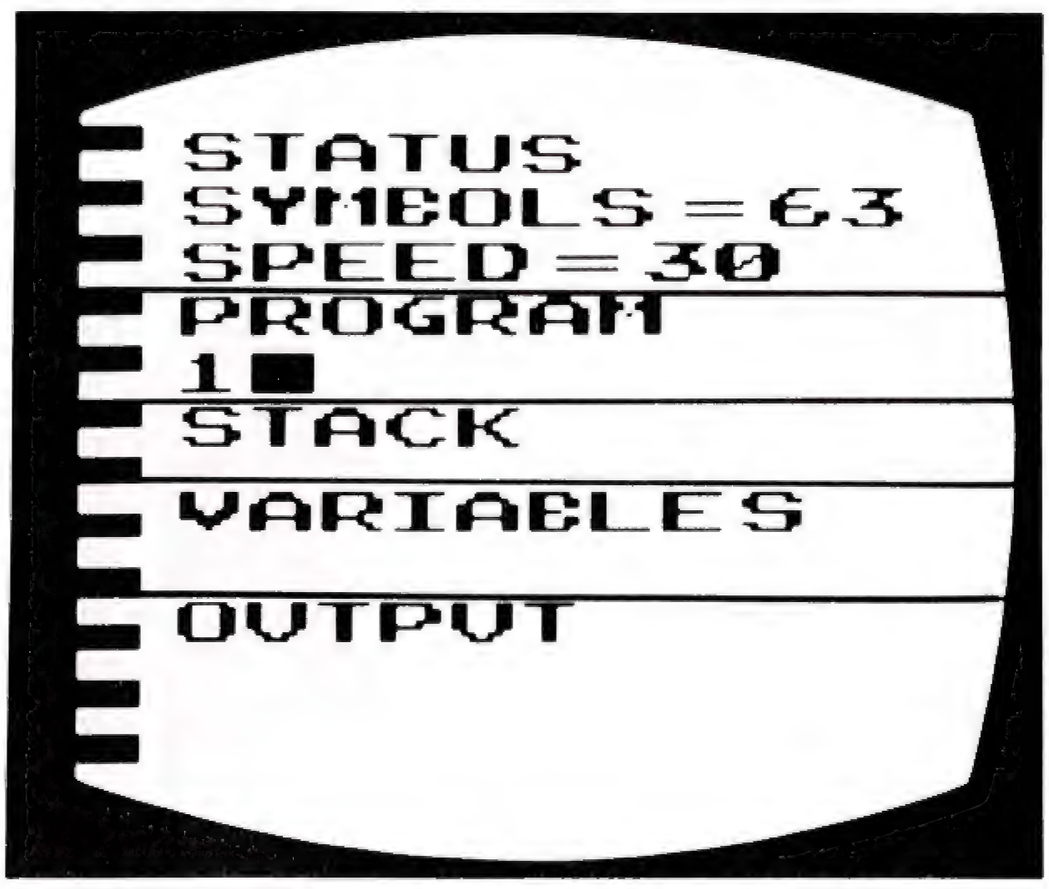
este sector también se presentará la velocidad de ejecución del programa.

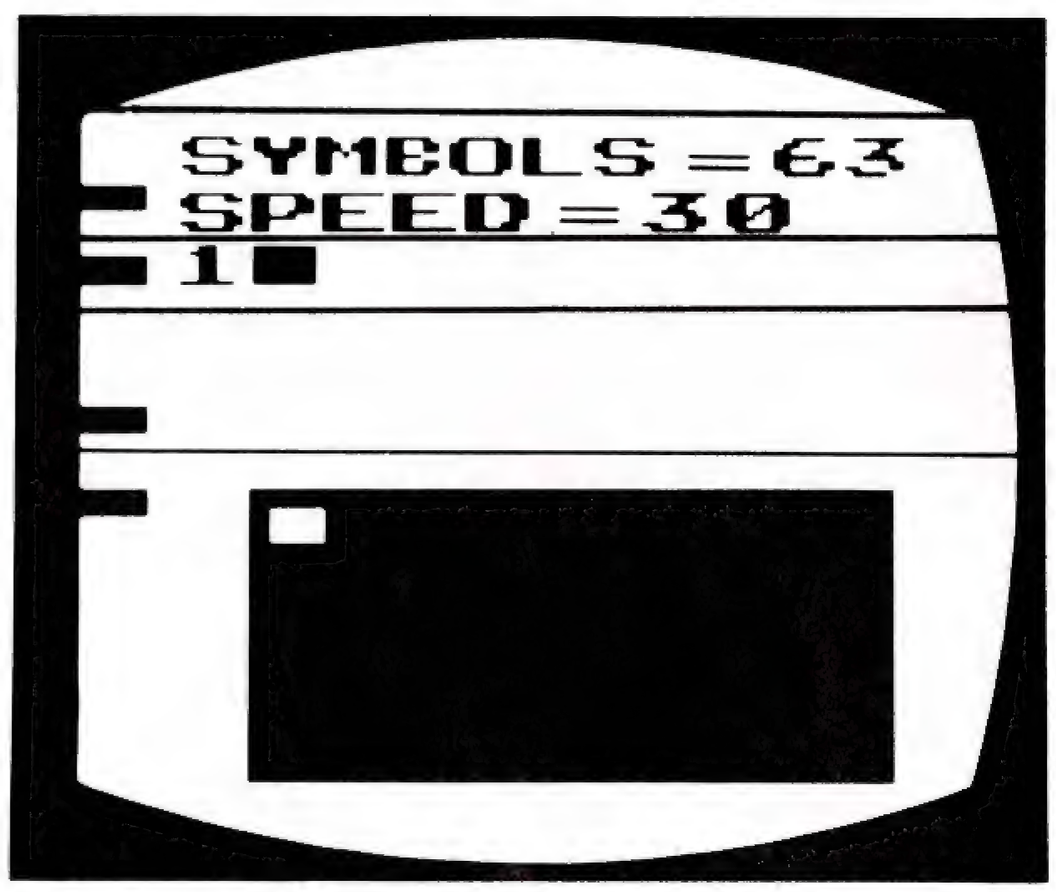
6. El sector gráfico **GRAPHICS** tiene dos cuadrados de color que se pueden mover bajo el control del programa.

Antes de iniciar la ejecución del programa es necesario fijar en la posición **b** el selector de grado de dificultad izquierdo **(left difficulty)**.

Esto permitirá visualizar en la pantalla la ubicación de los distintos sectores. En cambio, cuando el selector se fija en la posición **a** desaparecerán de la pantalla los nombres de cada sector y en cambio se presentará el sector de gráficos GRAPHICS.

El selector de grado de dificultad derecho **(right difficulty)** no interviene en este juego de programación.





EL CURSOR

Deslice el selector de dificultad izquierdo **(left difficulty)** a la posición **b**, apague la consola colocando el interruptor **power** en la posición **off** y a continuación vuelva a encenderla colocándolo en la posición **on**. En el sector de programación **PROGRAM** se presenta un rectángulo blanco, que es el cursor. Ahora ubique la tecla de cambio en el centro de la fila inferior de teclas en el comando insertado en el receptáculo marcado **LEFT CONTROLLER**, y oprímala cuatro veces. Esta acción hará cambiar el color del cursor de blanco a rojo, de rojo a azul a verde y de verde nuevamente a blanco.

El cursor se emplea para ingresar un programa. Cada uno de los cuatro colores de la tecla de cambio corresponde a las instrucciones en colores o ingresos efectuados mediante el teclado. El

modo fijado mediante el color blanco se emplea para ingresar instrucciones en la máquina, y los modos correspondientes a los colores restantes se emplean para ingresar símbolos en un programa.

TRASLADO DEL CURSOR DE UN SECTOR A OTRO

Asegúrese de que el selector de dificultad izquierdo **(left difficulty)** se encuentra fijado en la posición **b**, apague la consola y luego vuelva a encenderla. Oprima la tecla de avance **FORWARD** y el cursor se deslizará del sector **PROGRAM** al sector de la escala **STACK**. Vuelva a oprimir la tecla **FORWARD** y el cursor pasará del sector **STACK** al sector de las variables **VARIABLES**. Oprima nuevamente la tecla y el cursor pasará de **VARIABLES** al sector de salida **OUTPUT**.

Oprimiendo la tecla de retroceso **BACKWARD** puede hacer retroceder el cursor hasta el sector **PROGRAM**. Las teclas **FORWARD** y **BACKWARD** se pueden oprimir una vez para cada sector o bien se pueden mantener oprimidas hasta que el cursor llegue al sector deseado.

A continuación fije el selector de dificultad izquierdo **(left difficulty)** en la posición **a** y desaparecerán los rótulos de los sectores. Aparecerá una parte del sector gráfico **GRAPHICS**. Ahora haga pasar nuevamente el cursor por cada uno de los sectores. Observe que el cursor no entrará en el sector **GRAPHICS**.

BORRADO DE SECTORES DE LA PANTALLA

Fije el sector de dificultad izquierdo **(left difficulty)** en la posición **b** y apague la consola y luego vuelva a encenderla. En el teclado izquierdo hay una serie de comando (modo blanco) que corresponden a cada sector: **STATUS, PROGRAM, STACK, VARIABLES, OUTPUT** y **GRAPHICS**. Comencemos con el sector **STATUS**. Oprima una vez la tecla **STATUS** y el sector correspondiente se borrará de la pantalla, y el sector **PROGRAM** pasará a la parte superior de la pantalla.

Oprima la tecla **PROGRAM** y el sector correspondiente se borrará de la pantalla, y el sector **STACK** pasará a la parte superior de la pantalla. Oprima la tecla **STACK** y el sector correspondiente se borrará, y el sector **VARIABLES** pasará a la parte superior de la pantalla. Oprima la tecla **VARIABLES** y se borrará el sector correspondiente, y el sector **OUTPUT** pasará a la parte superior de la pantalla.

El sector **OUTPUT** se borra oprimiendo la tecla **OUTPUT**, lo que también presentará en la pantalla el sector **GRAPHICS**. Este último

sector se borra oprimiendo la tecla **GRAPHICS**. Ahora la pantalla no presenta ningún sector.

Oprima ahora la tecla **STACK** y reaparecerá el sector **STACK** en la pantalla. Borre el sector **STACK** y presente los sectores **OUTPUT** y **VARIABLES**. Puede presentar o borrar cualquier sector manteniendo el selector de dificultad izquierdo **(left difficulty)** en la posición **a** o en la posición **b**.

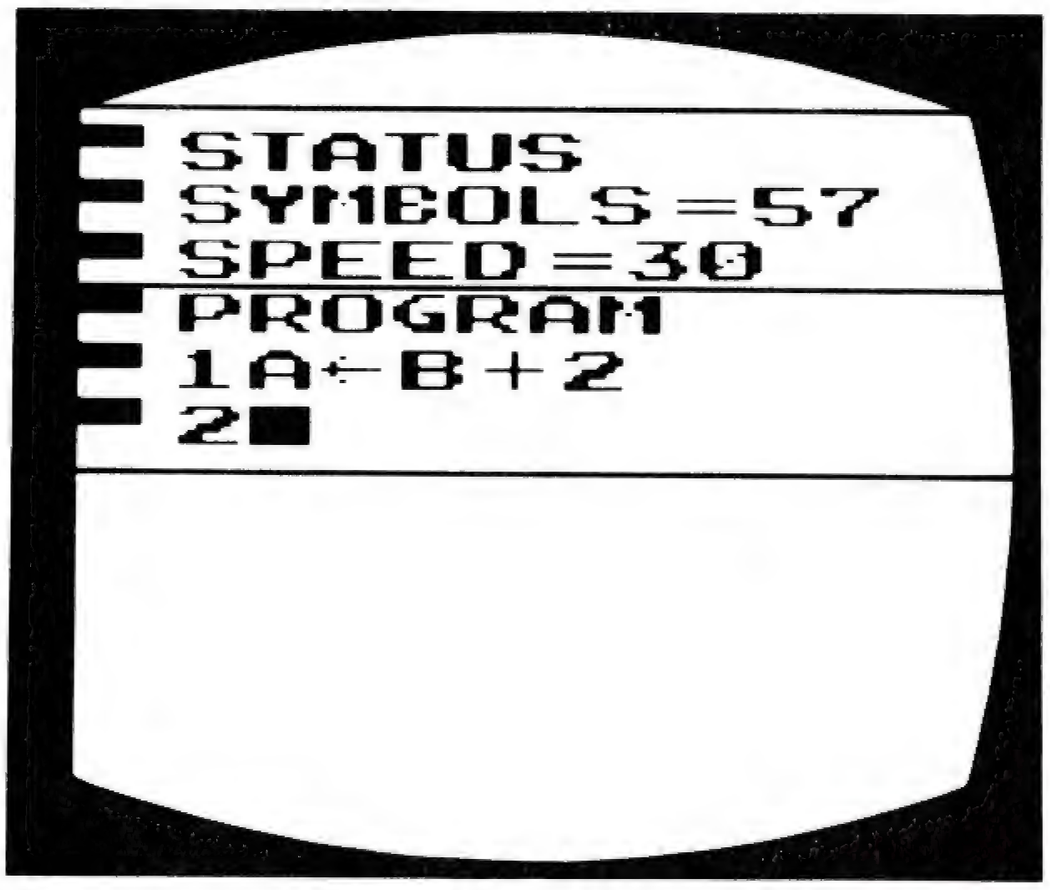
Es importante observar que la presencia o ausencia de los diversos sectores en la pantalla no afecta la ejecución de un programa.

EJECUCIÓN DE UN PROGRAMA

Hagamos un programa sencillo. Asegúrese que el selector de dificultad izquierdo **left difficulty** se encuentre fijado en la posición **b**, y luego apague la consola y vuelva a encenderla. (Otra forma de borrar un programa y reposicionar todos los valores consiste en deslizar hacia abajo el selector de juego **(game select)**. Al deslizar hacia abajo el conmutador de reposición **(game reset)** se borran todos los valores y se hace retornar el programa al comienzo sin que se borre su contenido.)

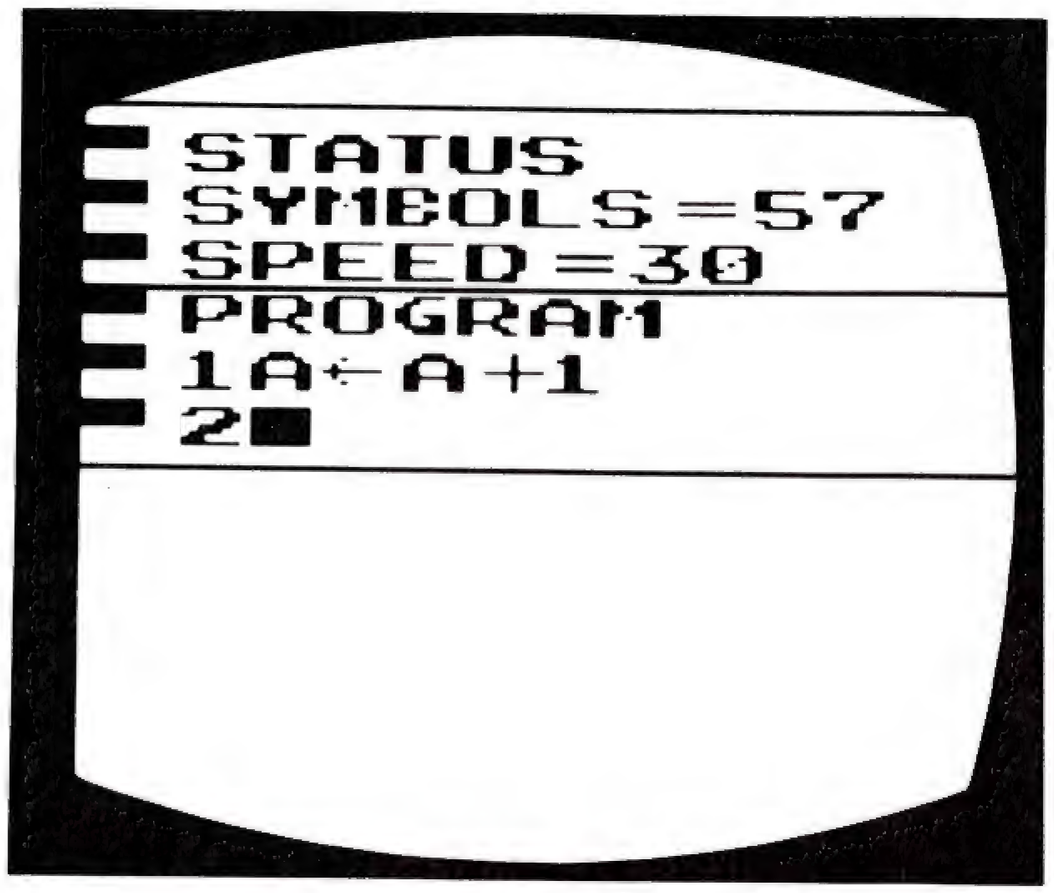
Borre de la pantalla los sectores **STACK, VARIABLES, OUTPUT** y **GRAPHICS**. El cursor estará en el modo blanco y a la derecha del número 1 en el sector **PROGRAM**. Cambie el cursor al modo azul y oprima la tecla **A**. La letra **A** aparecerá en la pantalla al lado del número **1**. (Cada línea está numerada, lo que hace posible ver dónde termina una línea y comienza otra.) A continuación cambie el cursor al modo rojo y oprima la tecla ←. Aparecerá una pequeña flecha junto a la **A**. Retorne al modo azul e ingrese la letra **B**. Ahora cambie el cursor al modo rojo, ingrese el signo + y el número **2**. **A** continuación fije nuevamente el modo blanco e ingrese la expresión **New Line** (nueva línea, nuevo renglón). Para iniciar una nueva línea en su programa, la computadora siempre debe estar fijada en el modo blanco.

La pantalla debe verse en esta forma:



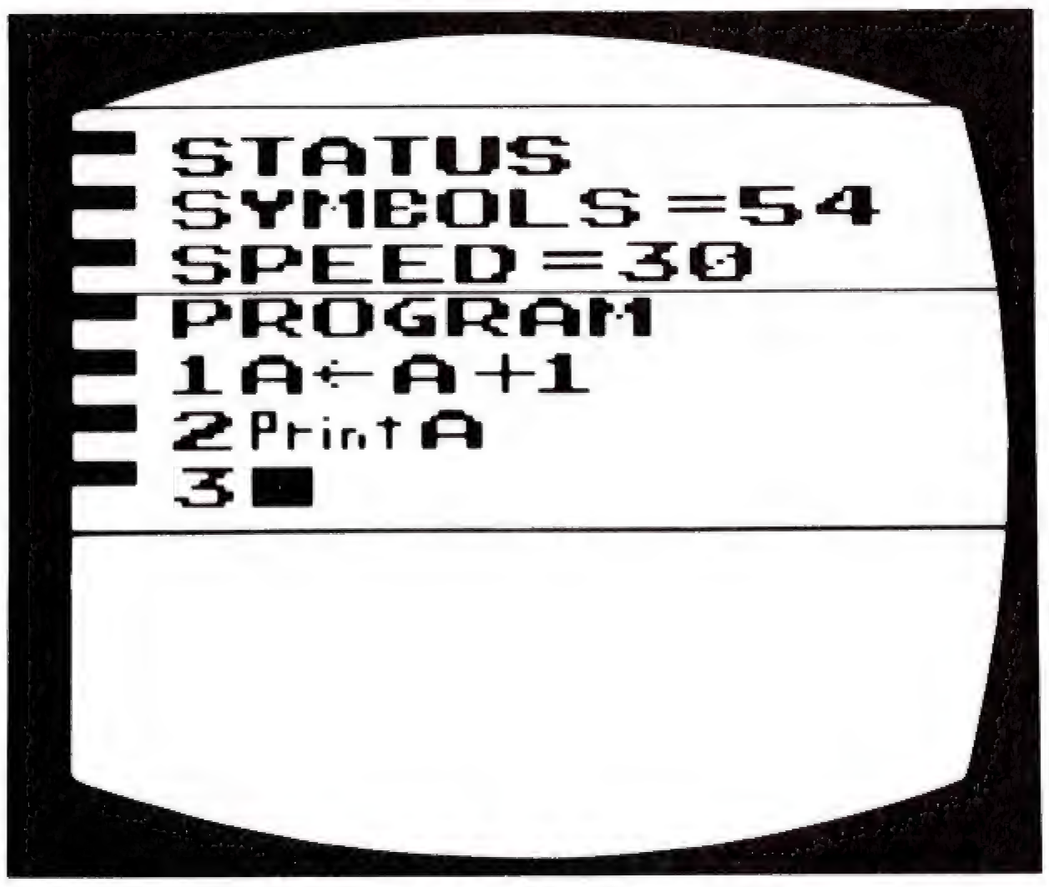
Si usted comete un error en el ingreso de los datos del programa, puede borrarlo de la pantalla oprimiendo la tecla **ERASE** (borrar), que no tiene ningún color, por lo que se puede utilizar cuando la tecla de cambio se encuentra en cualquier modo de color.

Hagamos un ejercicio de práctica utilizando la línea que acaba de ingresar en su programa. Oprima una vez la tecla **ERASE**. El cursor se deslizará de la línea 2 hasta la derecha del número 2 en la línea 1. Oprima nuevamente la tecla **ERASE** y el 2 se borrará del programa. A continuación fije la máquina en el modo rojo e ingrese el número 1. Cambie el cursor al modo blanco y oprima la tecla **BACKWARD** para deslizar el cursor hasta ubicarlo a la derecha de la **B** en el programa. Oprima la tecla **ERASE** y la letra se borrará del programa. A continuación ponga una **A** en su lugar (modo azul). En el modo blanco, emplee la tecla **FORWARD** para deslizar el cursor hasta el final de la línea 1, e ingrese la expresión **New Line** (nueva línea).



Observe que el cursor no debe estar sobre el símbolo que se desea borrar, sino inmediatamente a la derecha del mismo. Ahora la pantalla se verá en esta forma:

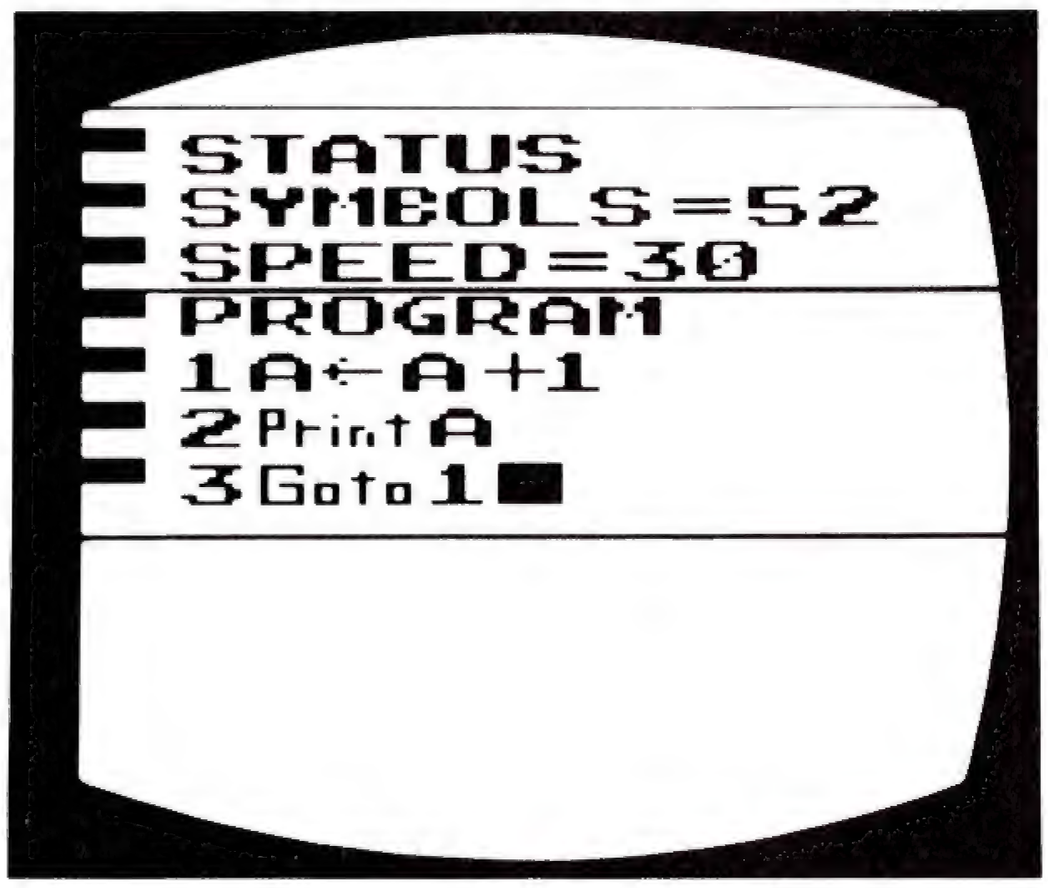
El cursor se encuentra inmediatamente a la derecha del **2** en la línea **2**. A continuación ingrese la expresión **PRINT** con la máquina en el modo verde. Luego, en el modo azul, ingrese la letra **A** y pase a una nueva línea. La pantalla se verá en esta forma:

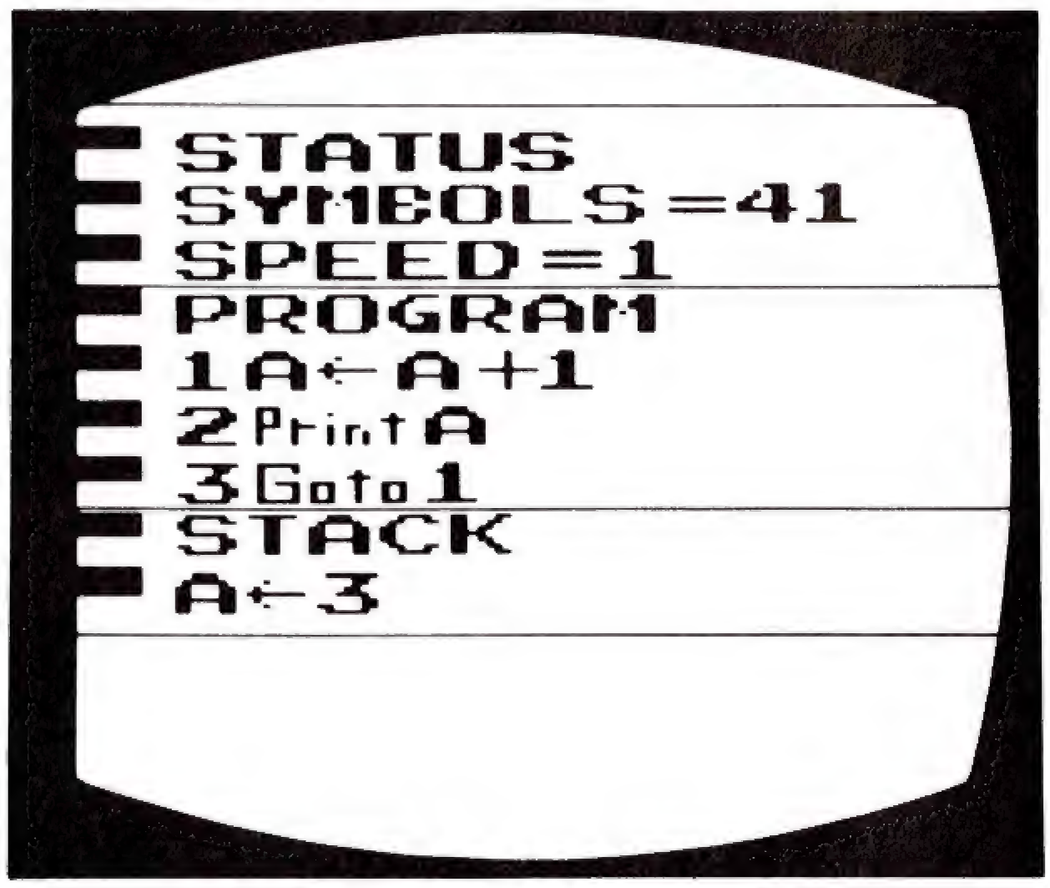


Observe que el sector **STATUS** muestra que se dispone de 52 bytes o símbolos de memoria. La velocidad **SPEED** se ha fijado en **30 (SPEED = 30)**. Coloque el cursor en el modo blanco y oprima la tecla **SLOWER** (más lento). Observe que cada vez que oprime esta tecla disminuye la velocidad. Oprima la tecla **FASTER** (más rápido) y la velocidad aumentará.

Antes de comenzar la ejecución del programa, ingrese **SPEED = 1**. A continuación oprima la tecla **STACK**. Oprima dos veces la tecla **RUN/HALT** y la máquina comenzará a ejecutar el programa. Usted puede observar la ejecución de cada parte del programa.

Ahora, con el cursor en el modo verde, ingrese la expresión **GOTO** (vaya a), y a continuación ingrese el número **1** en el modo rojo. Antes de proseguir volvamos a mirar la pantalla:





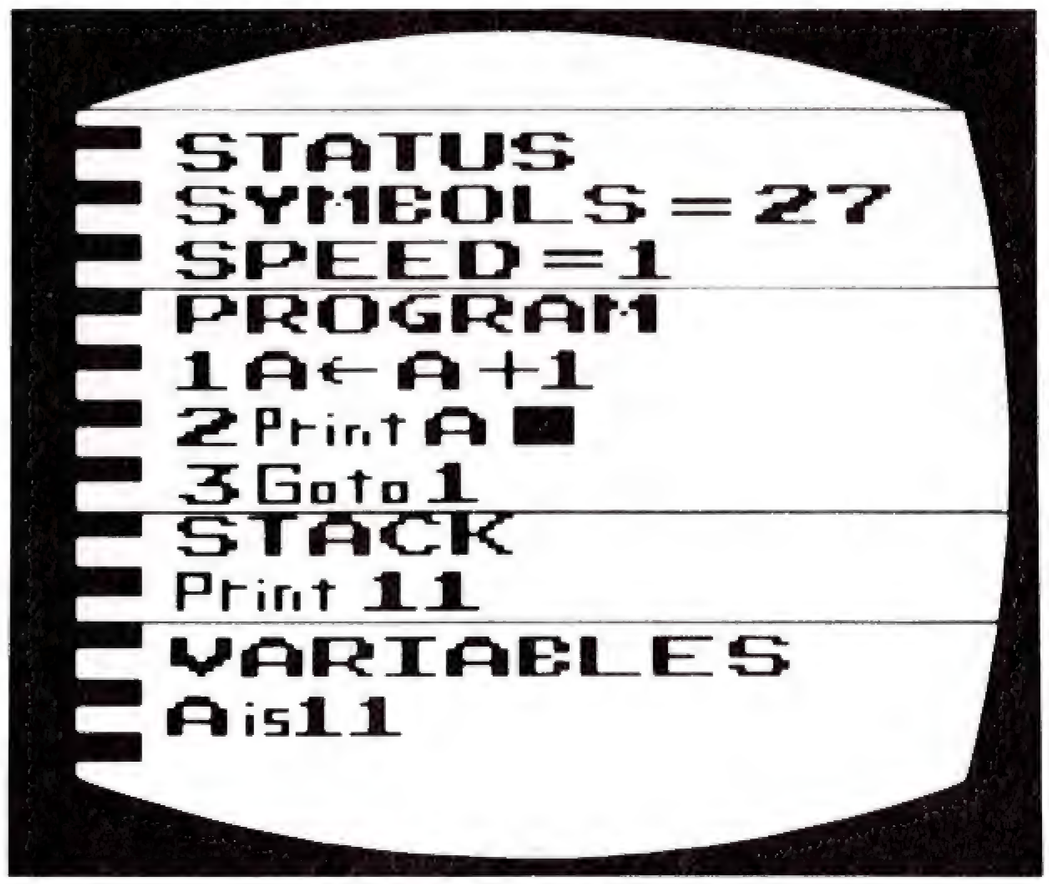
Para detener la ejecución de un programa, oprima la tecla **RUN/HALT**. Como dijimos antes, al accionar el conmutador de reposición **(game reset)** se borran todos los valores de un programa, pero sin que se borre el programa mismo, y el programa vuelve a su forma inicial.

Analicemos el programa paso por paso a medida que es ejecutado. Cambie la velocidad **SPEED** a **60**. Fije el cursor en el modo blanco y oprima la tecla **STEP**, lo que hará avanzar el programa paso por paso cada vez que oprima la tecla.

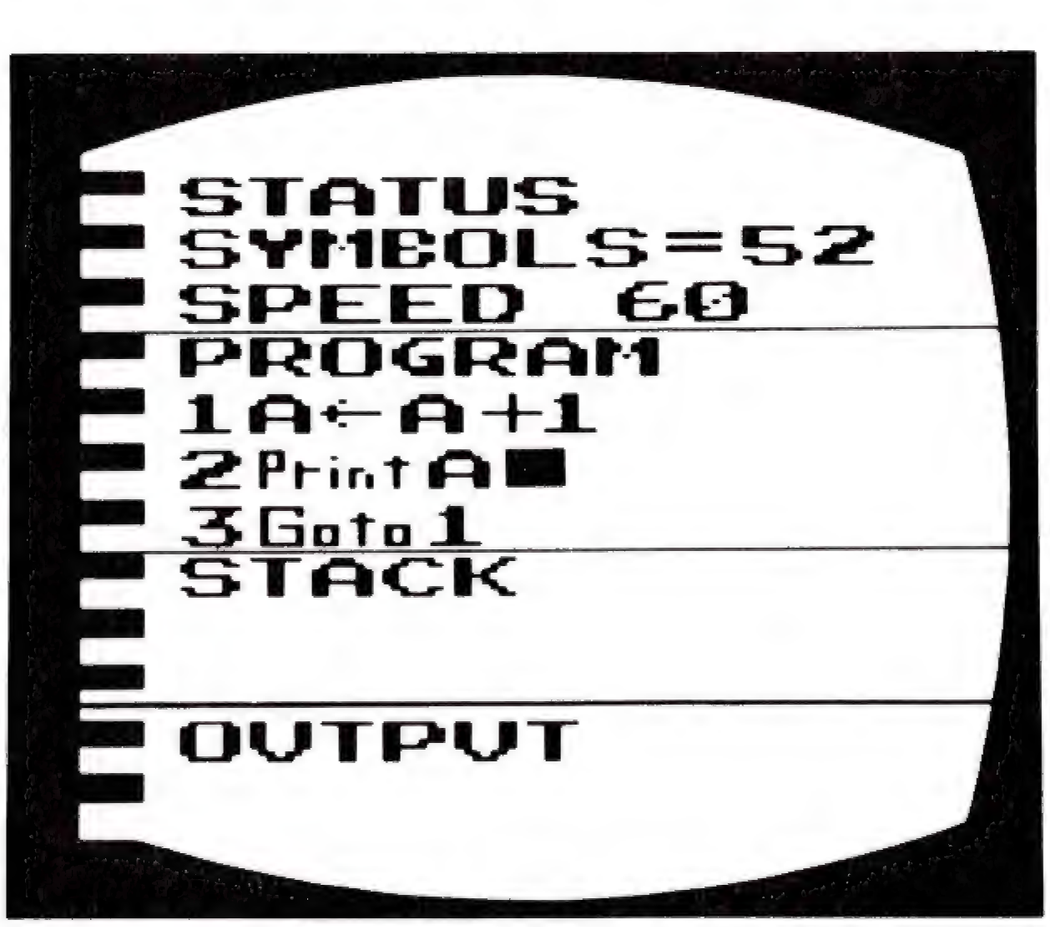
La línea del programa aparece en esta forma: **A ← A + 1**. La máquina la lee así: **A** se convierte en **A + 1**. Observe el sector de la escala **STACK** al cambiar la línea **1**. En la línea **2** usted había ingresado la instrucción **Print A** (imprimir A). Esto significa que usted desea que la máquina presente el valor de **A** (según ha sido determinado por la línea 1) en el sector de salida **OUTPUT**.

Ya veremos en qué forma se realiza esto. La línea **3** instruye a la máquina **Goto 1** (Vaya a 1). La ejecución del programa pasa a la línea 1 para encontrar un nuevo valor para **A**. Cada vez que se ejecute el programa encontrará un nuevo valor para **A**, presentará ese valor y luego regresará a la línea **1**. La máquina seguirá ejecutando el programa en esta forma hasta que se use toda la memoria. La cantidad de memoria que queda disponible aparece en el área de símbolos **SYMBOLS** del sector de estado **STATUS**.

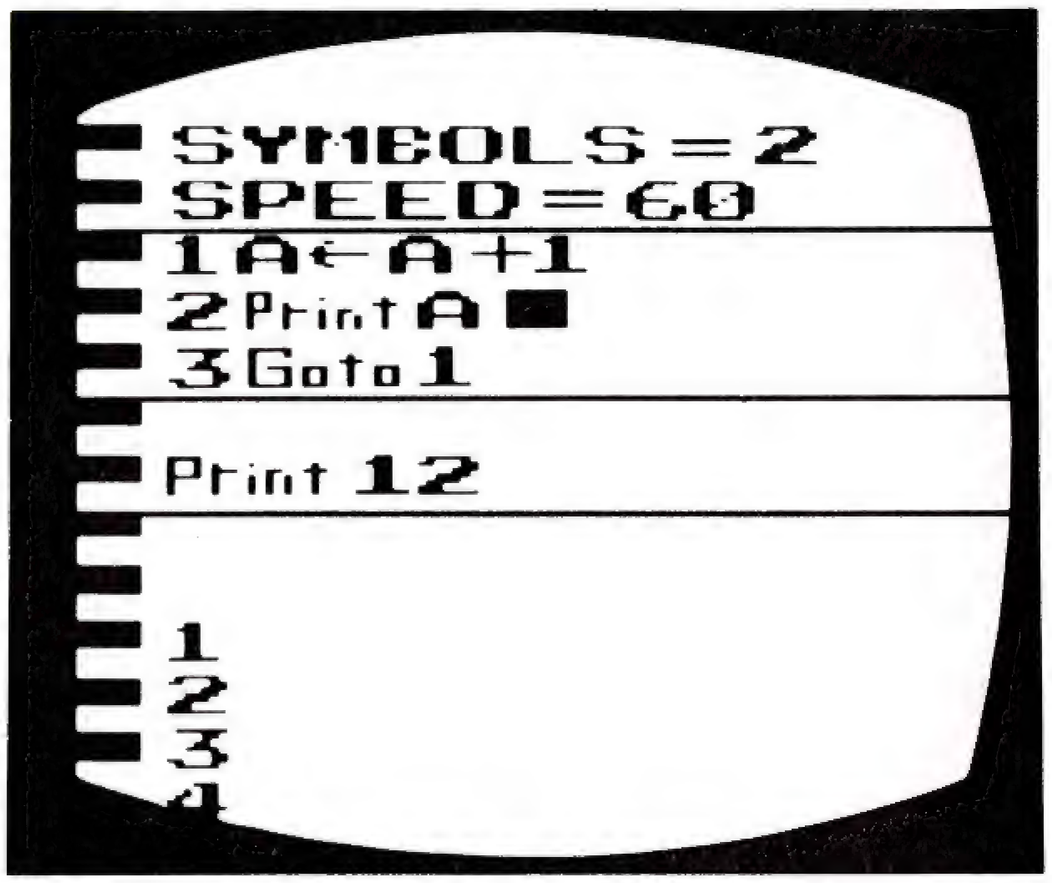
A continuación muestre en la pantalla el sector de las variables **VARIABLES**. Cambie la velocidad **SPEED** a 1 y borre los valores del programa accionando el conmutador de reposición de juego **(game reset)**. Oprima la tecla **RUN/HALT** y observe mientras la máquina ejecuta el programa. La máquina mostrará el valor actual de **A** en cada paso del programa. La pantalla se verá en esta forma:



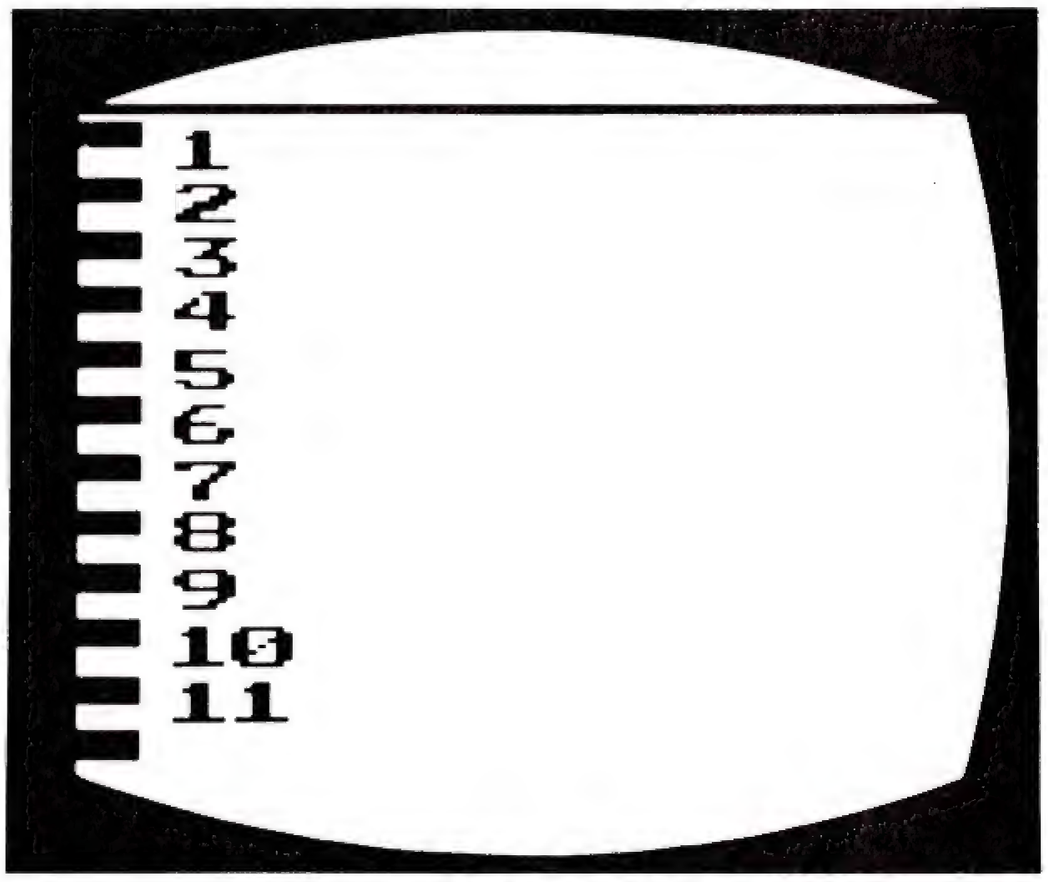
Pare la ejecución del programa y borre los valores **(game reset)**. Borre el sector **VARIABLES** y presente en su lugar el sector de salida **OUTPUT**. Cambie la velocidad **SPEED** a 60. Ahora la pantalla aparecerá así:



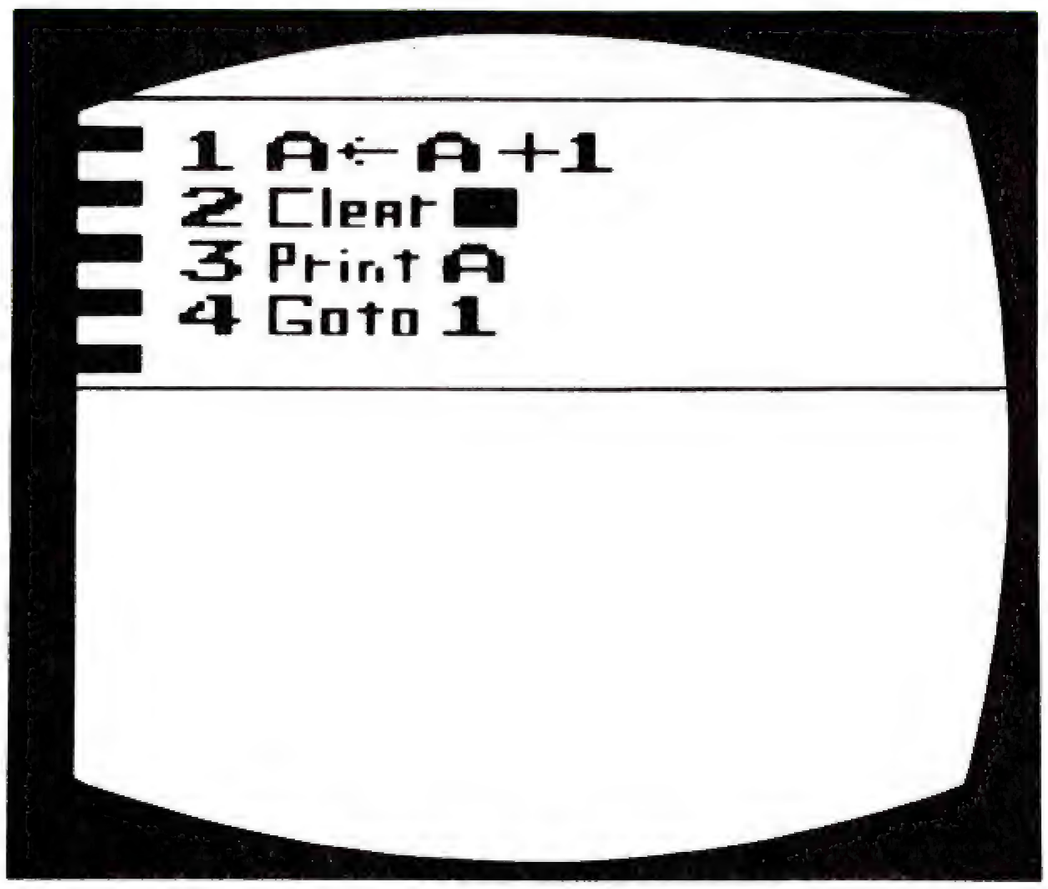
A continuación oprima la tecla **RUN/HALT** e inicie la ejecución del programa. En la línea **2** se da la instrucción **PRINT A** (presentar el valor de **A**). Cuando la ejecución llegue a este paso, en el sector **OUTPUT** de la pantalla se presentará el valor de **A**. Observe el área de símbolos **SYMBOLS** en el sector de estado **STATUS**. Aunque todavía la línea 1 aparece en el sector de salida **OUTPUT** en la pantalla, la máquina todavía sigue presentando el valor cambiante de **A**. Ahora fije el selector de dificultad izquierdo **left difficulty** en la posición **a**, y la pantalla se verá en esta forma:

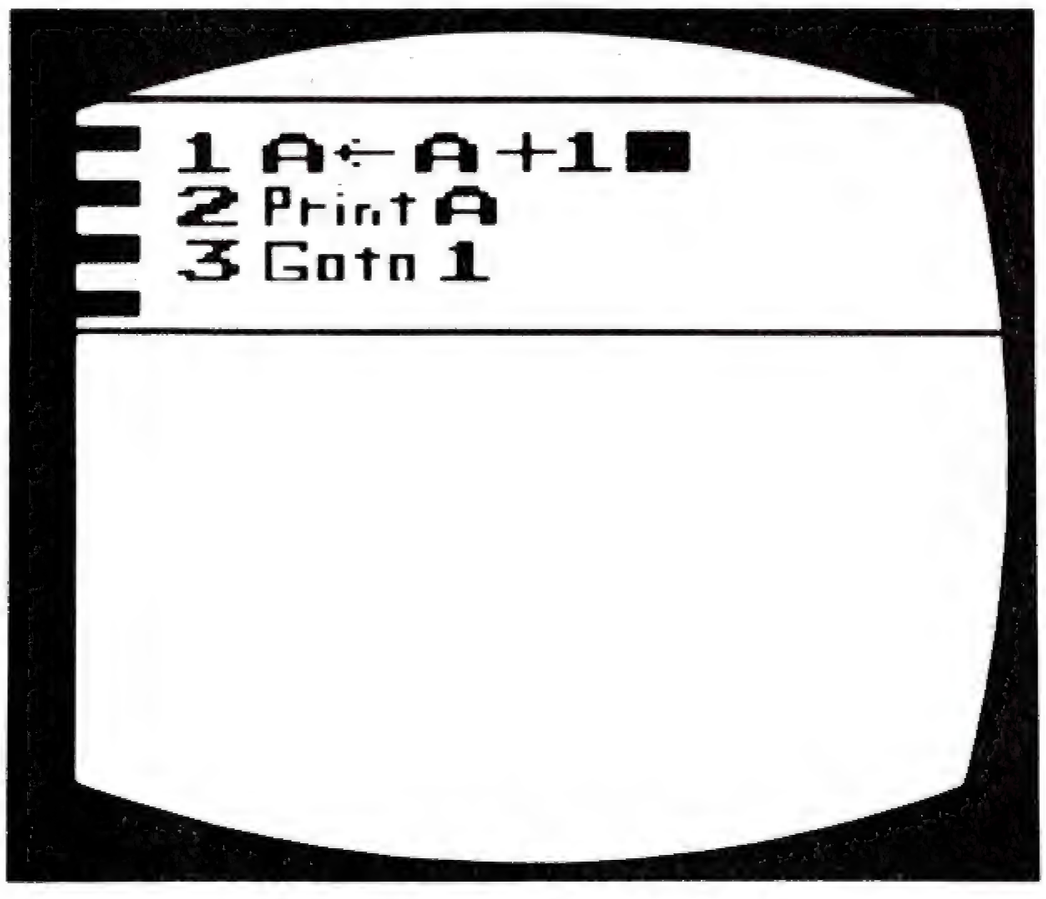


¿Cómo se podría presentar más el sector de salida **OUTPUT** en la pantalla? Elimine de la pantalla los sectores **STATUS, PROGRAM** y **STACK**. Ahora la pantalla se verá así:



Supongamos que no queremos recargar el sector de salida **OUTPUT** del programa. ¿Qué se debe hacer? Pare la ejecución del programa y borre los valores con el conmutador de reposición **(game reset)**. Elimine el sector **OUTPUT** de la pantalla y presente en cambio el sector de programación **PROGRAM**. Con la tecla de avance **FORWARD** deslice el cursor hasta el final de la línea **1**. La pantalla presentará este aspecto:





Ahora oprima la tecla de nueva línea **(NEW LINE)** para ingresar una nueva instrucción. Fije el modo verde e ingrese la expresión **CLEAR** (borrar), y la pantalla se verá en esta forma:

Presente en la pantalla los sectores **STACK, VARIABLES** y **OUTPUT**. Deje el selector de dificultad **(left difficulty)** en la posición **a** e inicie la ejecutión del programa. Cuando el programa encuentra la instrucción de borrar **(CLEAR)** en la línea **2**, borra el valor de **A** en el sector de salida **OUTPUT** y a continuación presenta el valor siguiente de **A** en la línea **3**.

Este juego de programación básica **BASIC PROGRAMMING** funciona solamente con números de dos dígitos, por lo que al llegar a 99 el programa comenzará nuevamente a presentar el valor de **A** como igual a 0 (=0).

EMPLEO DE LA FUNCIÓN NOTE (NOTA MUSICAL)

Cada vez que el programa almacene un número en **NOTE** (modo rojo), se escuchará una nota de la escala musical por el parlante del televisor.

Hagamos algunos ejercicios de programación para demostrar lo dicho. Si usted tiene algún programa en su Computer Video System, deslice hacia abajo el selector de juego **(game select)** en la consola.

A continuación ingrese lo que sigue:

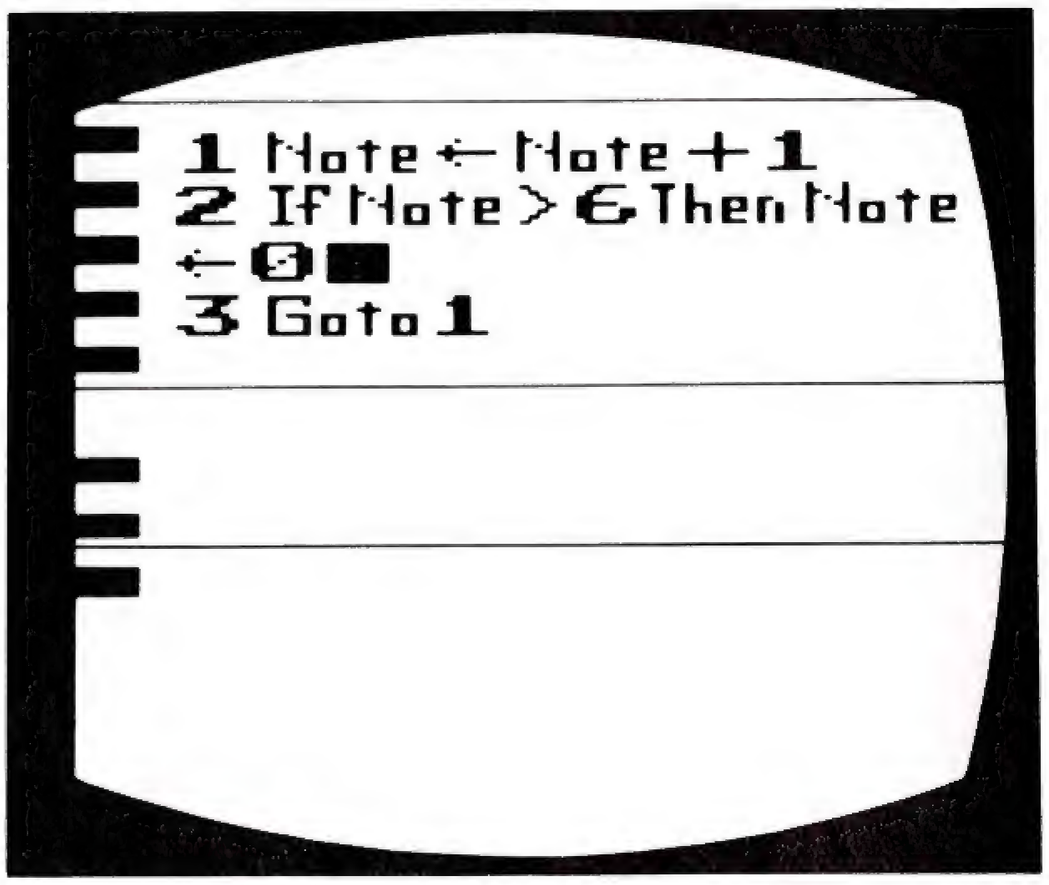
1 Note ← Note + 1
2 Goto 1

Ahora inicie la ejecución del programa. Haga más lenta la ejecución y observe lo que sucede en el sector de la escala **STACK**. Observe, además, que el programa presentará el valor de **NOTE** en el sector **VARIABLES**.

Pare el programa e inserte una nueva línea **2**. Para llevarlo a cabo, fije el modo blanco, deslice el cursor al final de la línea **1** y oprima la tecla de nueva línea **(NEW LINE)**. La que antes era línea **2** será ahora línea **3**.

2 If Note > 6 Then Note ← 0

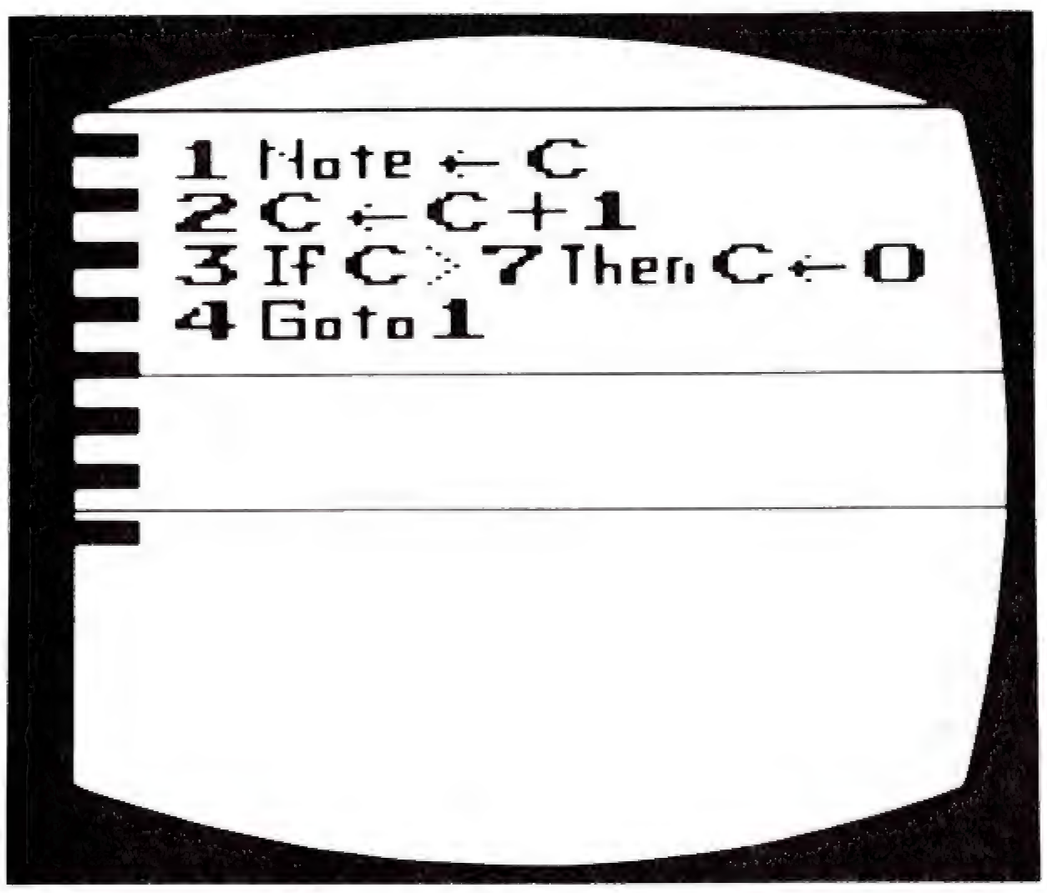
La pantalla se verá en esta forma: (En caso de que se hayan eliminado los sectores **STACK** y **VARIABLES**.)



Las instrucciones **IF** (si) y **THEN** (entonces) ordenan al programa lo siguiente: si **(IF)** sucede algo, entonces **(THEN)** debe hacer algo diferente. En este caso, IF **Note** tiene un valor de más de 6, **THEN** será necesario cambiar Note a cero (0). Inicie la ejecución del programa y observe su desarrollo en el sector de la escala **STACK**.

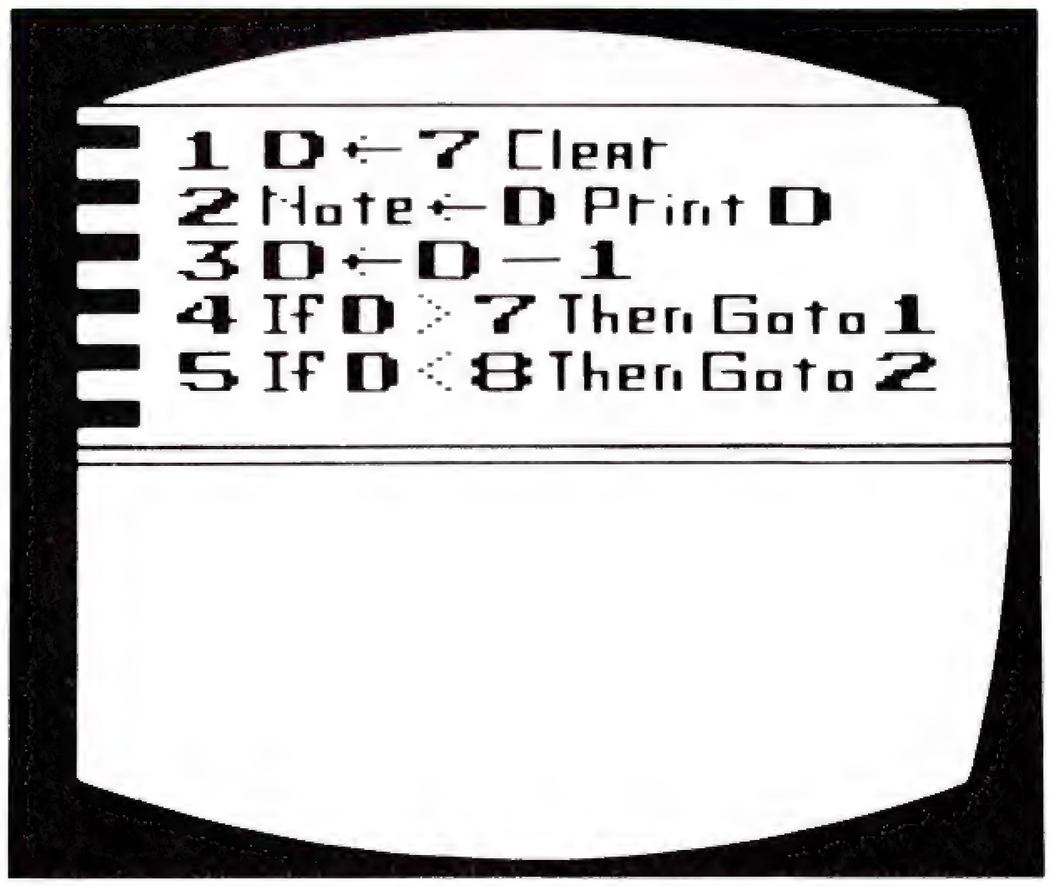
Observe que al llegar a **7** el valor de **Note** se hace mayor que (>) 6 y el programa cambia el valor a cero (0).

Ahora haremos otro programa para obtener un resultado idéntico pero por otro camino. Ingrese lo siguiente:



Ejecute el programa. Observe que se obtienen los mismos resultados, excepto que las notas están regularmente espaciadas. Si desea que el programa le dé el valor de **C** en el sector de salida **OUTPUT**, ingrese **"Print C"** (presentar el valor de **C**) y **"Clear"** (borrar) como instrucciones en cualquier parte del programa. Para disminuir el parpadeo en el sector **OUPUT** mientras el programa presenta el valor pedido, ingrese una coma (modo verde) después del valor en la línea de presentación, **Print C**.

Puede emplear cualquier letra del alfabeto como nombre de una variable en su programa. El programa que sigue tiene todas las funciones claves aprendidas hasta ahora, y utiliza casi toda la memoria disponible. Después de ingresar este programa, sáquelo de la pantalla, presente los sectores **STATUS, STACK** y **OUTPUT**, y ejecútelo.



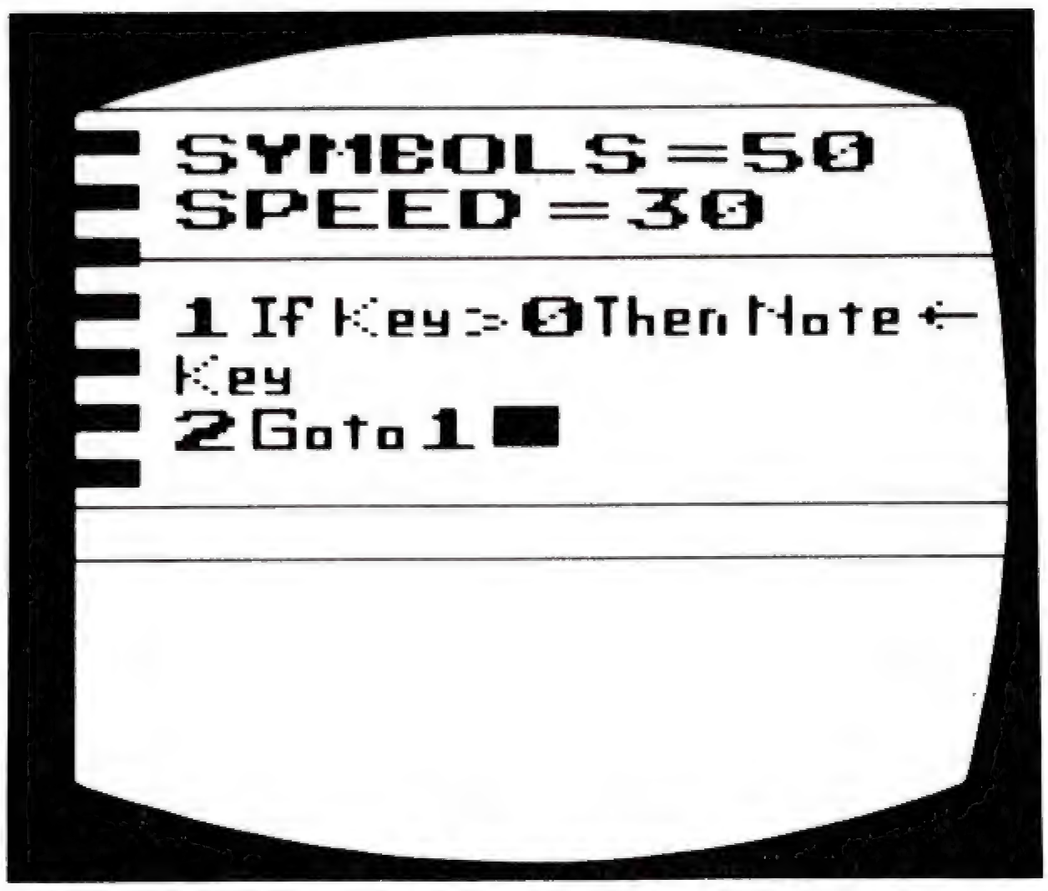
En este programa hemos incluido dos instrucciones **IF/THEN**. Si no se satisfacen las condiciones establecidas en la primera instrucción (línea 4), el programa pasará a la línea siguiente (línea 5). Dependiendo de la memoria disponible, puede haber varias instrucciones entre dos instrucciones **IF/THEN**.

En algunos casos podemos ingresar dos instrucciones en una sola línea, como en el caso de las líneas **1** y **2** en el programa anterior. En esta forma se puede ahorra algo de memoria para usarla más adelante en el programa.

EMPLEO DE LAS FUNCIONES KEY (INGRESO) Y PRINT (PRESENTACIÓN)

La función **KEY** (ingreso) se emplea para ingresar una variable mientra el programa está en ejecución. El programa evaluará **KEY** y lo reemplazará por un número que usted ingresa mediante el teclado del lado derecho del comando. Si no ingresa ningún número, el programa leerá **KEY** como cero (**0**).

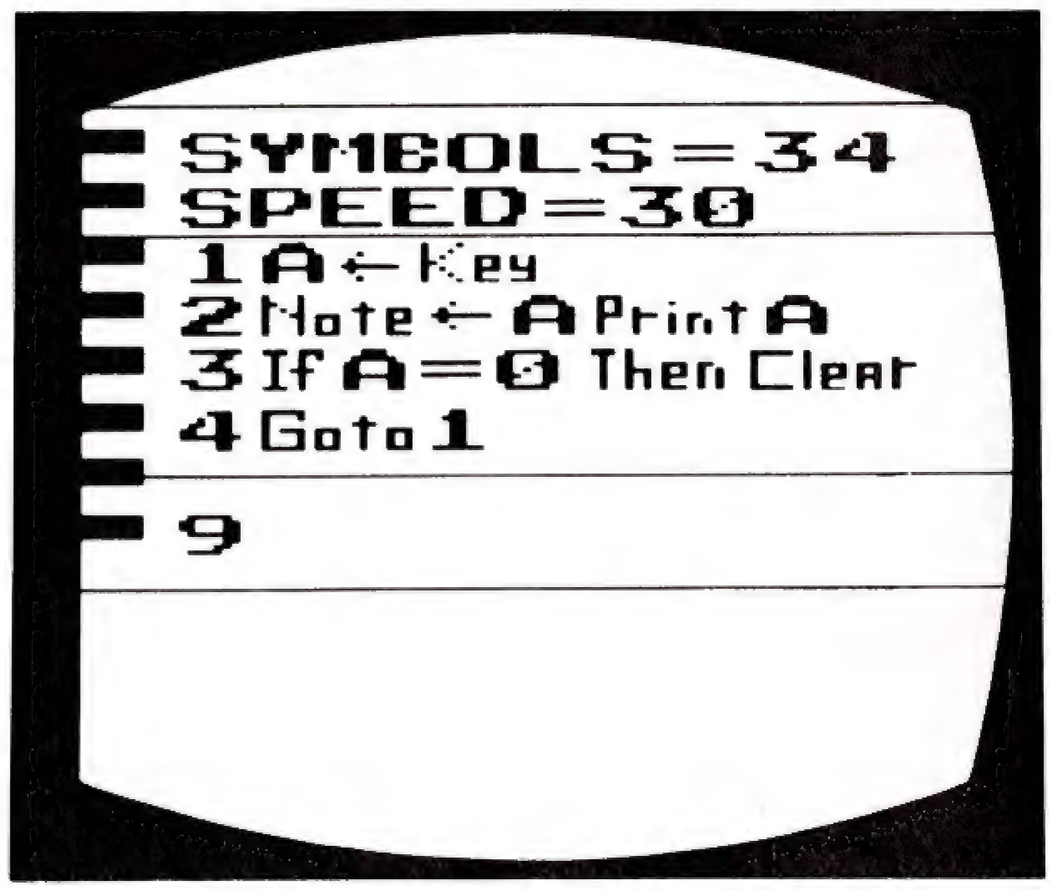
Éste es un programa sencillo que emplea la función **KEY**:



Inicie la ejecución del programa y oprima algunas de las teclas en el lado derecho del comando. Con algo de práctica podrá interpretar una melodía.

La función **KEY** también se puede utilizar para programar en el sector gráfico **GRAPHICS**, como lo veremos.

Ejercítese con este programa empleando las funciones **KEY**, **NOTE** y **PRINT**:

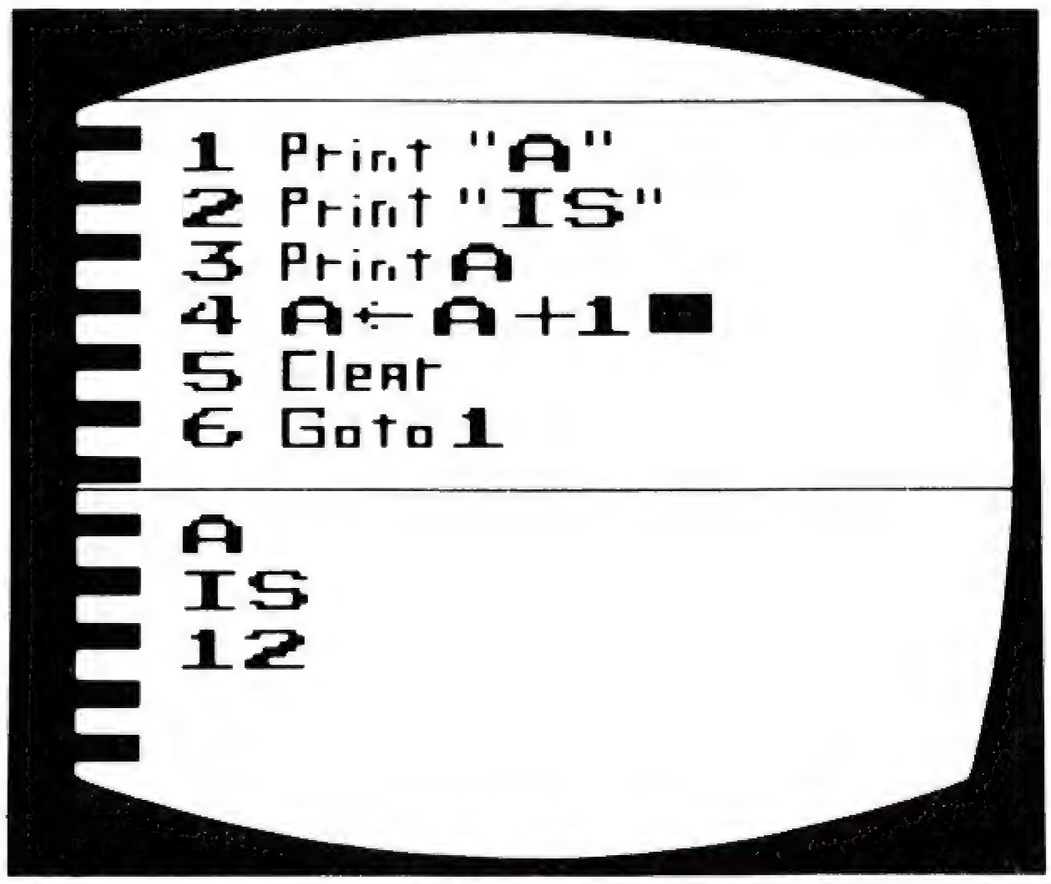


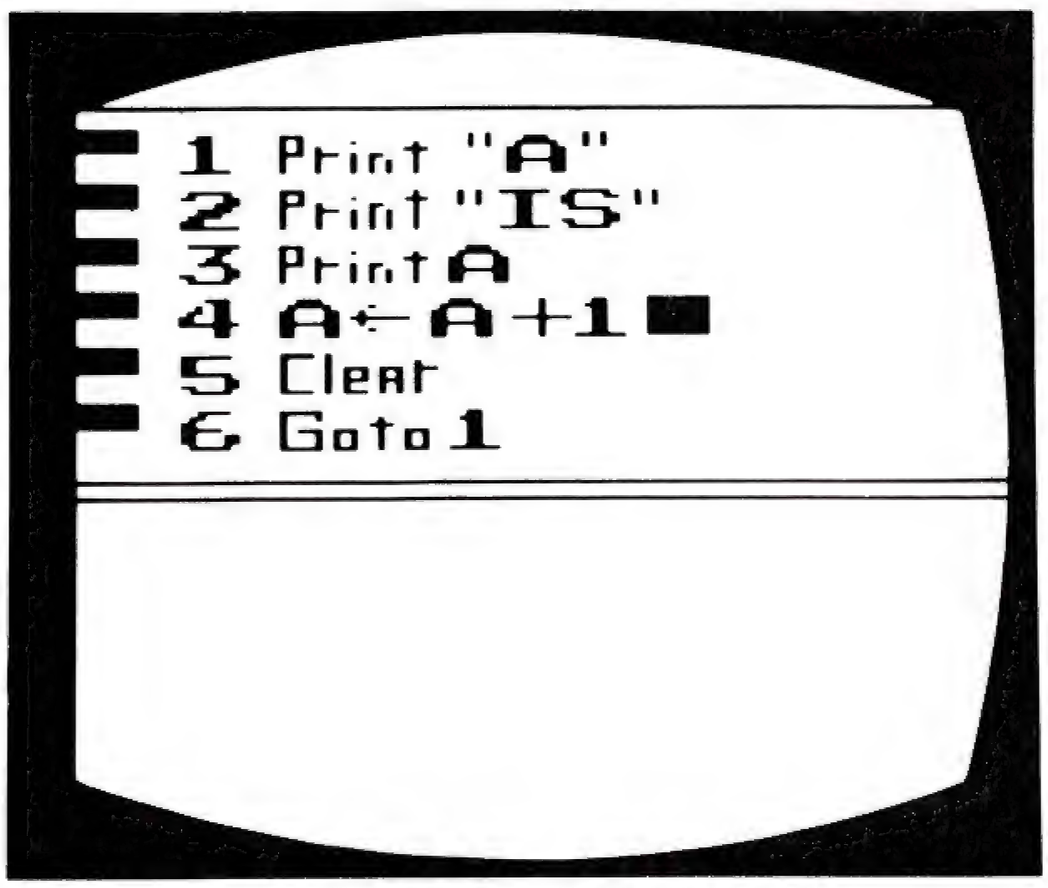
Observe el desarrollo de este programa en el sector de salida **OUTPUT**. Fíjese que cuando ingresa un número mediante el teclado del lado derecho del comando, el programa interpreta un tono musical y presenta el número ingresado en el sector de salida **OUTPUT**.

Introduzcamos ahora una modificación menor en el programa. En la línea **2** ingresemos una coma (,) despues de la instrucción **Print 1**, e iniciemos la ejecución del programa. Al ingresar la coma en el lugar indicado, se instruye al programa que se desea tener presentadas en una sola línea tantas variables como sea posible.

EMPLEO DE LA FUNCIÓN DE PRESENTACIÓN PRINT

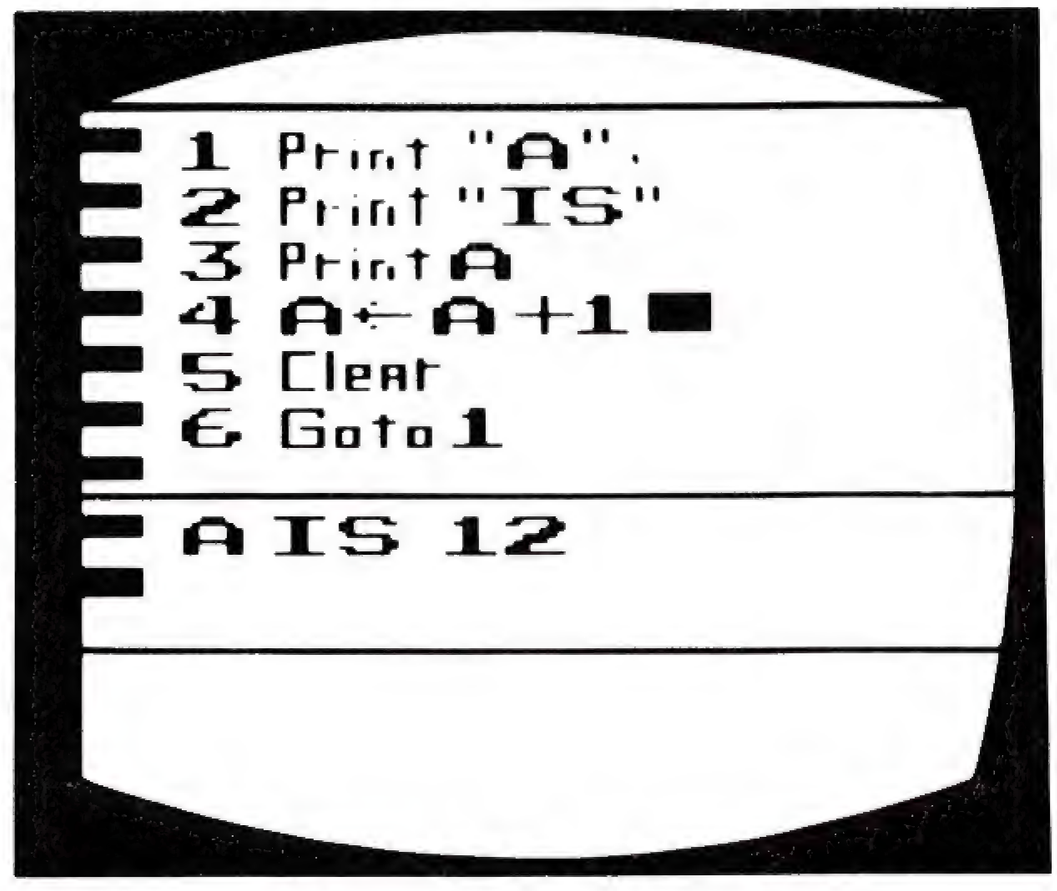
Como y explicamos en programas anteriores, usted puede emplear la función de presentación **PRINT** para ordenarle al programa que presente el valor de una variable en el sector de salida **OUTPUT**. La función **PRINT** también se puede utilizar para ordenar al programa que presente palabras en la pantalla. Las palabras que se desea presentar debeningresarse entre comillas (''). Fije la velocidad **SPEED** en **8** e ingrese el siguiente programa:



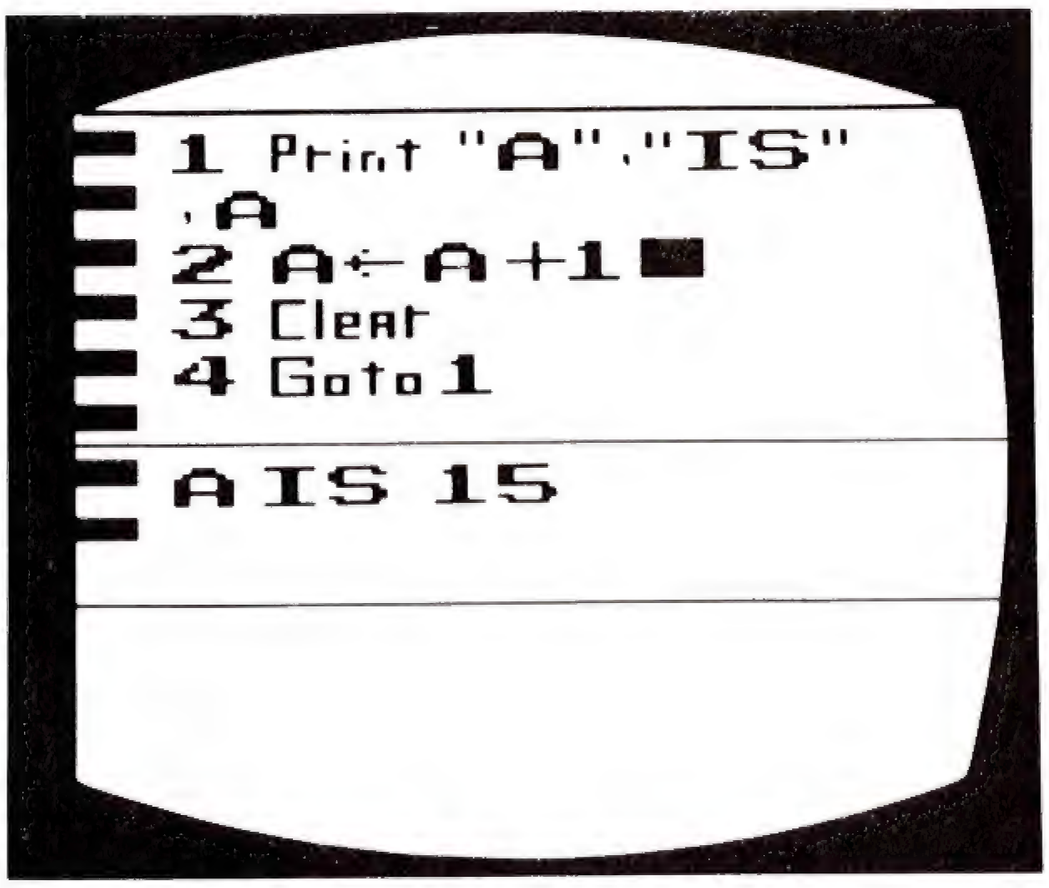


Ahora ejecute el programa y observe lo que sucede en el sector de salida **OUTPUT**. El programa presenta las palabras que se han ingresado pero dispuestas en columna.

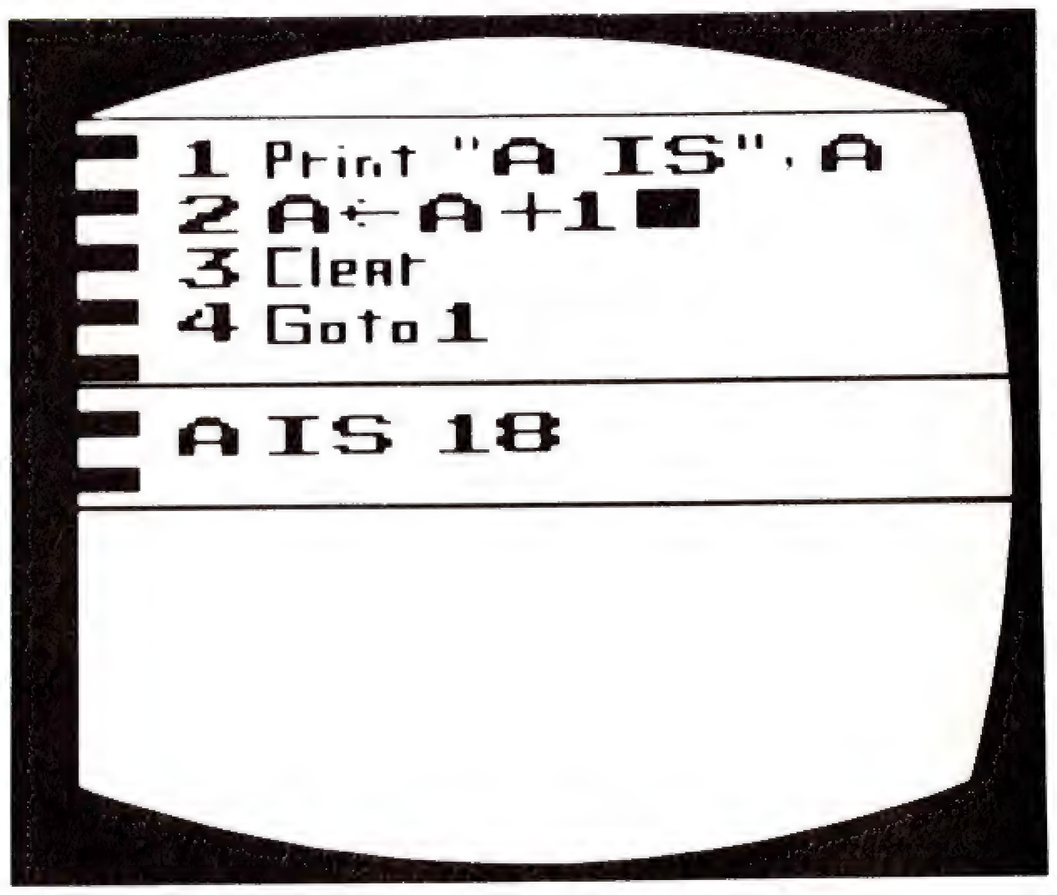
Cambiemos levemente el programa ingresando una coma (,) al final de las líneas **1** y **2**. Ahora el programa aparece así:



La coma (,) en la línea ha ordenado al programa que los datos de la línea **2** deben ser presentados en el sector de salida **OUTPUT** en la misma línea, tal como la instrucción de la línea **3** debe seguir a la línea **2**. Estas instrucciones se pueden acortar si se cambia el programa en esta forma:



Es posible acortar más aún el programa:



Al escribir el programa en esta forma también se ahorrará espacio de memoria.

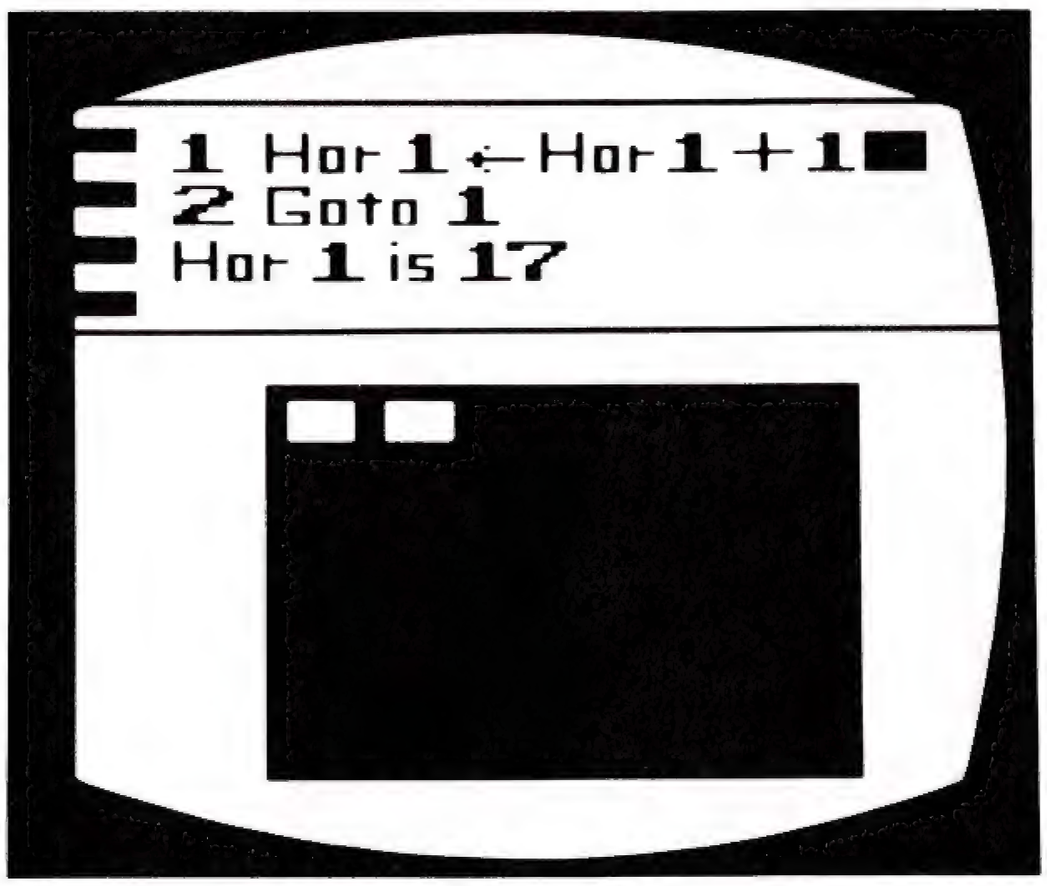
EMPLEO DEL SECTOR GRAPHICS GRÁFICO

El sector gráfico **GRAPHICS** es el campo rectangular de color azul que aparece en la pantalla. A primera vista parecería que contiene un solo cuadrado rojo en el ángulo superior izquierdo. El cuadrado rojo en realidad está superpuesto sobre un cuadrado blanco, y ambos se pueden mover independientemente en el campo controlado por el programa.

Para mover los cuadrados es necesario que usted cambie sus coordenadas. El cuadrado rojo es el objeto número 1. Su coordenada horizontal está representada en el teclado por **Hor 1**, y su coordenada vertical por **Ver 1. Hor 2** y **Ver 2** son las coordenadas del objeto número 2, que es el cuadrado blanco.

Ambos objetos o cuadrados parten del ángulo superior izquierdo del campo azul, pero únicamente el cuadrado rojo es visible. El ángulo superior izquierdo es la posición cero (o) u origen. Cuando no se definen (se les da valor) las variables **Hor 1, Ver 1, Hor 2** y **Ver 2**, tienen valores de cero (0) (por ejemplo: **Hor 1 ← 10**), el objeto afectado saltará a la posición correspondiente en el campo azul, cuando se ejecute el programa.

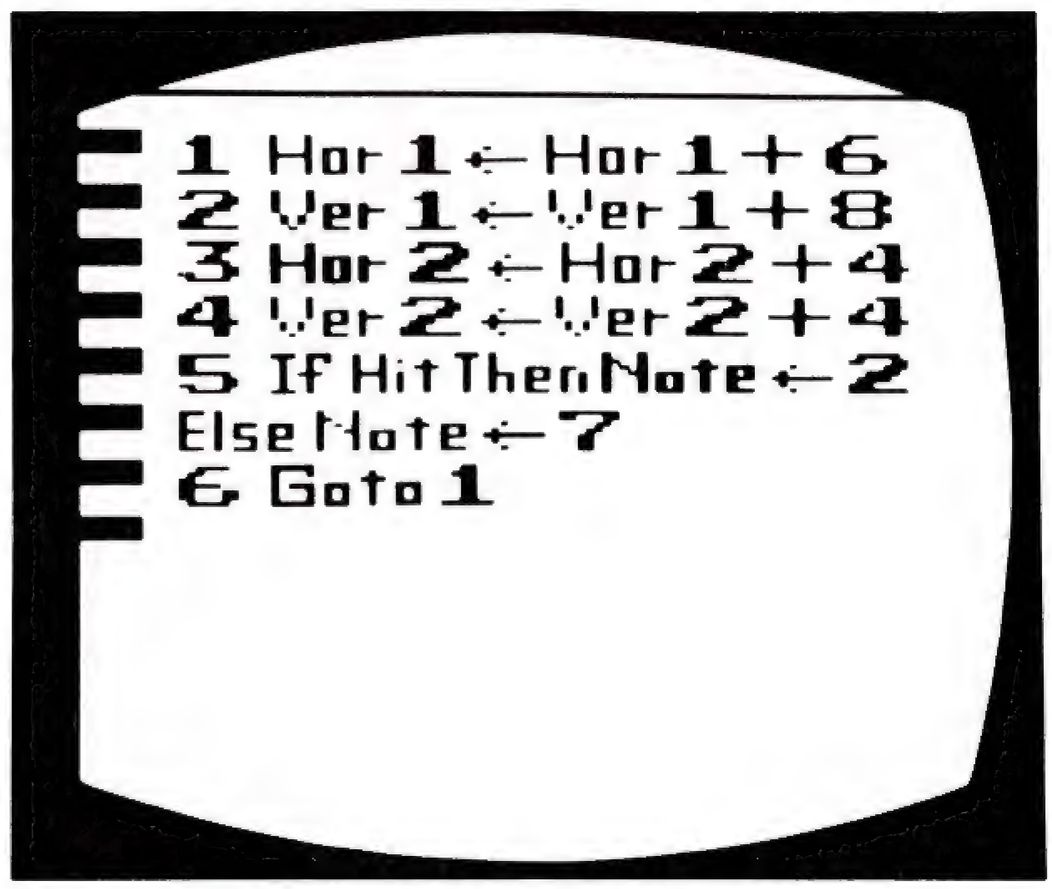
Ingrese el siguiente programa:



Este programa moverá el cuadrado rojo a la derecha, a razón de un espacio horizontal por vez. Ejecute este programa y verá que el cuadrado rojo se traslada lentamente hacia la derecha. Observe el sector de variables **VARIABLES** y verá aumentar el valor **Hor 1**. Seguirá aumentando hasta llegar a **99**, que es el número mayor permitido, y luego regresará a cero (**0**). Cuando **Hor 1** llegue a **99**, el cuadrado rojo habrá llegado al lado derecho opuesto del campo azul. En este punto, el cuadrado rojo retorna al lado izquierdo del campo, y **Hor 1** adquiere el valor de cero **(0)**.

Las coordenadas horizontales (**Hor 1** y **Hor 2**), entonces, pueden tener valores que van de **0** a **99**. Las coordenadas verticales (**Ver 1**y **Ver2**) también tienen valores que van de **0** a **99**; el **0** es la posición en la parte superior del campo azul, y el **99** es la posición en la parte inferior del mismo.

Ingrese el programa siguiente:



Saque el programa de la pantalla y asegúrese de que el sector gráfico **GRAPHICS** esté bien visible. Fije la velocidad en **60** y ejecute el programa. Este programa muestra una forma de mover los cuadrados en el campo. También revela la

forma de emplear las functiones **HIT** y **ELSE**.

En la línea **5** el programa instruye a la máquina que haga sonar la nota **2 IF** (si) los cuadrados **HIT** (chocan), lo que hacen periódicamente. Para que se produzca un **HIT**, los cuadrados deben ocupar las mismas coordenadas. De otro modo **(ELSE)** la máquina es instruida a que toque la nota **7**, lo que hará hasta que los cuadrados choquen o coincidan en la misma posición.

(Borre la instrucción **ELSE NOTE** ← **7** de la línea **5**, y la máquina tocará la nota **2** cuando los cuadrados chocan, y no tocará ninguna otra nota.)

EMPLEO DE LA FUNCIÓN MOD

Mod es un operador aritmético, semejante al operador de división (÷). El juego de programación básica **BASIC PROGRAMMING** emplea división con enteros, lo que significa que opera únicamente con números enteros, sin dejar residuo. Por ejemplo, ¿cuánto es 14 ÷ 5? Usted dirá: 2 y 4/5, o bien 2 y un residuo de 4. En este juego de programación básica, 14 ÷ 5 = 2. Aunque 5 cabe 2 veces en 14 (lo que es igual a 10) y queda un residuo de 4, BASIC emplea sólo números enteros, por lo que la respuesta que dará a la división mencionada será 2.

La computadora dará la misma respuesta para 12 ÷ 4 que para 13 ÷ 4. En ambos casos la respuesta será 3, puesto que 4 está contenido 3 veces en 12 como en 13. La máquina no reconocerá el residuo de 1 en el caso de 13 ÷ 4.

Veamos ahora la función **Mod**. **Mod** recibe el residuo de la división del primer número por el segundo. De modo que **14 Mod 5** es **4**. Cinco está contenido **2** veces en **14** (lo que es igual a **10**) y queda un residuo de **4**. **Mod** recibe el residuo, y por lo tanto es igual a **4**.

¿Cuánto es **13 Mod 4**? La respuesta es **1**, puesto que **1** es el residuo de la división de **13** por **4** (lo que es igual a **12**). ¿Cuánto es **12 Mod 4**? En este caso **Mod** es igual a **0**, puesto que **4** está contenido exactamente **3** veces en **12**, sin dejar ningún residuo.

Normalmente, en matemáticas la división por cero es indefinida. En **BASIC PROGRAMMING**, la división por 0 da un resultado de 0; de modo que 5 ÷ 0 = 0.

ORDEN DE PRIORIDAD DE LOS OPERADORES

En una ecuación, los operadores aritméticos (+, −, x, ÷, Mod, etc.) se resuelven en un orden de prioridad establecido. **BASIC**

PROGRAMMING emplea el orden de prioridad que sigue:

1. x ÷ (mázima prioridad)
2. + −
3. Mod
4. =
5. ← (mínima prioridad)

EMPLEÓ DE PARÉNTESIS

Cuando se resuelve una ecuación, el empleo de paréntesis (modo verde, lado izquierdo del teclado) da prioridad de cálculo a cualquier número contenido entre los paréntesis.

Por ejemplo, en el caso de la ecuación que sigue:

A 5 + 3 x 2 Mod 7

el primer paso es: 3 x 2 = 6
el segundo paso es: 5 + 6 = 11
el tercer paso es: 11 Mod 7 = 4

A = 4

Sin embargo, esta misma ecuación con paréntesis, se resuelve en forma diferente y también el resultado en el caso de A es diferente.

A (5 + 3) x 2 Mod 7
primer paso: 5 + 3 = 8
segundo paso: 8 x 2 = 16
tercer paso: 14 Mod 7 = 2

A = 2

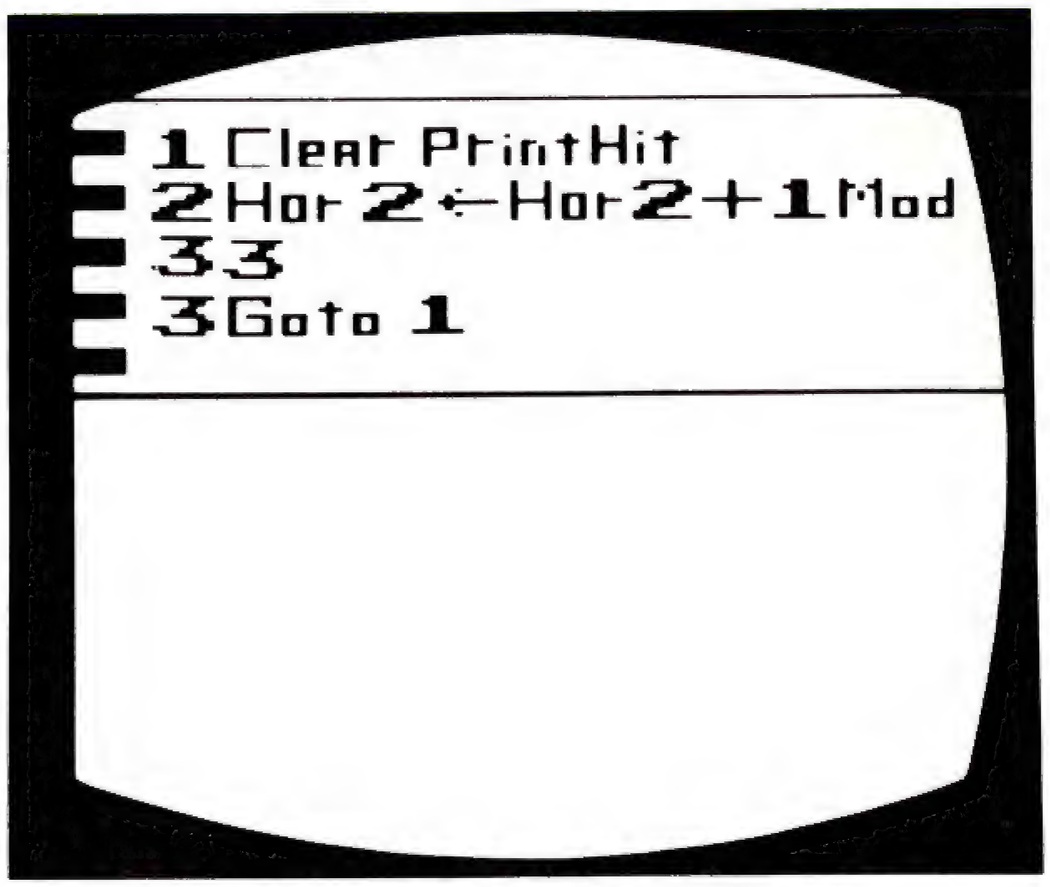
PROGRAMAS DE MUESTRA

A continuación damos algunos programas de muestra. Esto le ayudará a ver la amplia gama de posibilidades con las que puede trabajar. Recuerde que debe prestar cuidadosa atención a la forma como una instrucción específica (**IF, THEN, HIT, ELSE, PRINT, KEY,** etc.) afecta un programa. También debe recordar que una instrucción **KEY** puede depender de un ingreso efectuado desde el lado derecho del teclado para que el programa funciones con éxito.

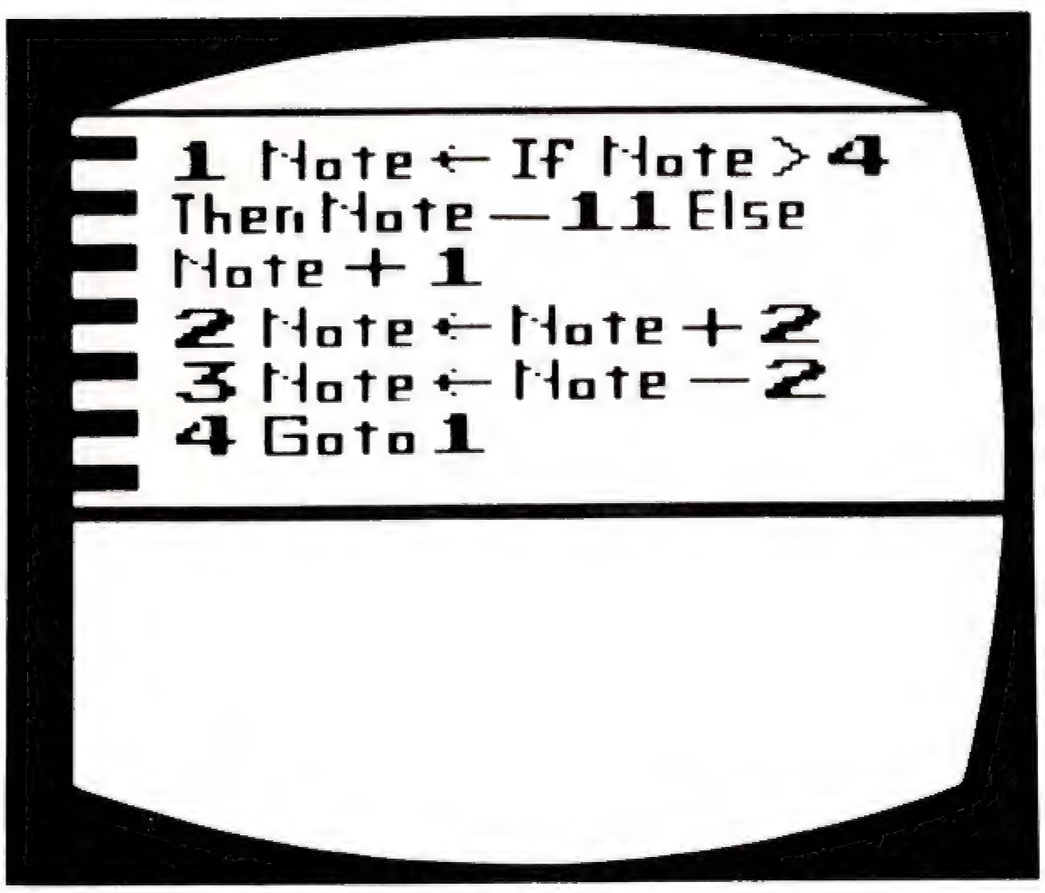
En caso de sentirse confundido por un programa en particular, pare la ejecución del ese programa, reposiciónelo al comienzo y ejecútelo paso a paso empleando la función **STEP**. Al observar el desarrollo del programa en el sector de escala **STACK** y al ver la forma como la computadora trabaja en cada paso, podrá comprender lo que está sucediendo y por qué está sucediendo.

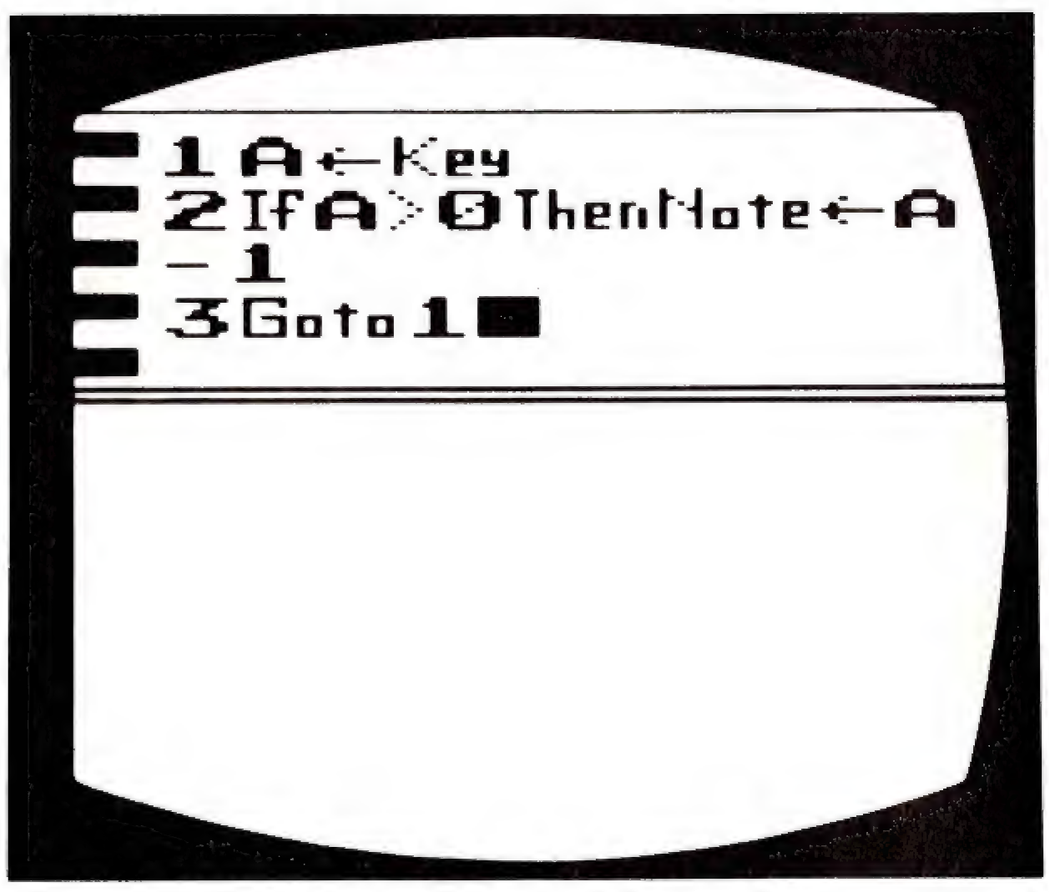
Una vez que haya ejecutado los programas y se haya familiarizado con los principios básicos de programación, estará en condición de tratar de crear algunos programas propios.

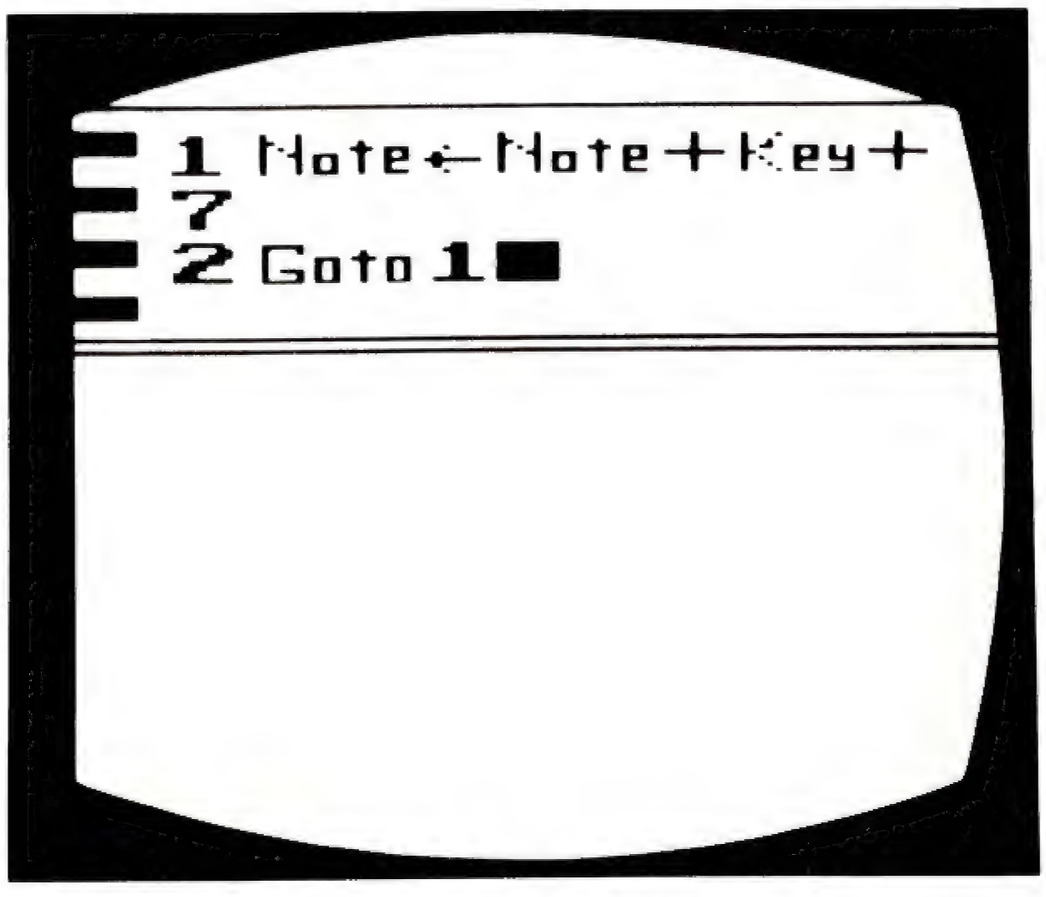
SUPERPOSICIÓN

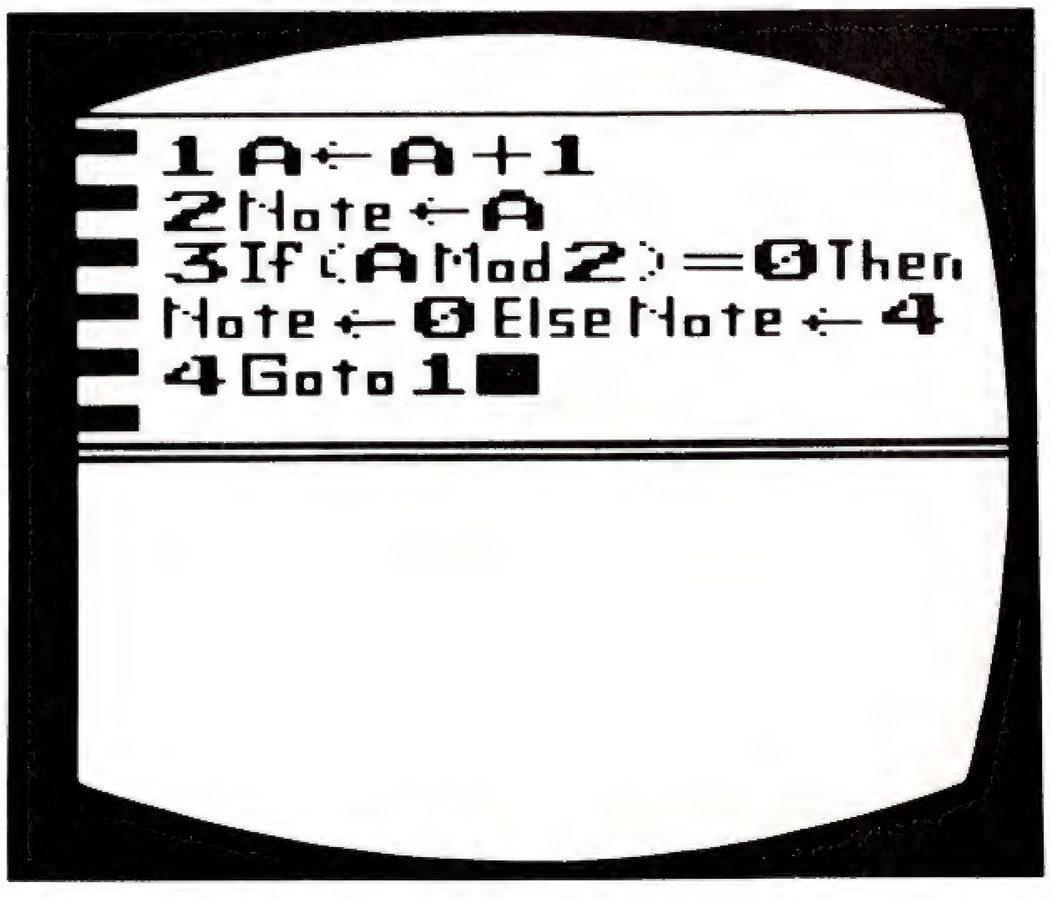


MÚSICA









PROGRAMA DEL RELOJ

Cuando ejecute el programa del reloj, presente en la pantalla solamente el sector de salida **OUTPUT**. La velocidad se puede fijar en 30 ó 60.

```
1 Clear
2 A ← A + 1 Mod 60
3 If A = 0 Then B ← B + 1
Mod 60
4 Print B . A
5 Goto 1
```

PONG® GAME

sin sonido

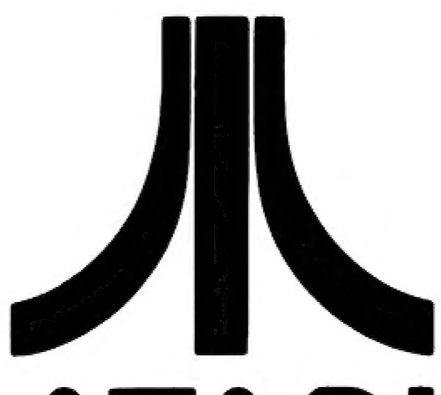
```
1 Hor 2 ← Hor 2 + Key
2 If Ver 1 > 90 Then
Ver 1 ← 88
3 If Hit Then Ver 1 ← 3
4 Ver 1 ← Ver 1 + If
Ver 1 Mod 2 Then 8 Else
32
5 Hor 1 ← Hor 1 + 7
6 Goto 1
```

PONG® GAME
(PELOTA Y PALETA)

```
1 Hor 1 ← Hor 1 + Key
2 Hor 2 ← Hor 2 + 8
3 Ver 2 ← Ver 2 - 3
4 If Hit Then Ver 2 ← 93
. Note ← 7
5 Goto 1
```

BASIC PROGRAMMING

BASIC PROGRAMMING



ATARI®

A Warner Communications Company

ATARI, INC., Consumer Division, P.O. Box 427, Sunnyvale, CA 94086, U.S.A.

CO16973-20 REV. 1

PRINTED IN U.S.A.